KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Analysis and Design of Masking Schemes for Secure Cryptographic Implementations

Oscar Reparaz

Promotor:
Prof. dr. ir. Ingrid Verbauwhede
dr. Benedikt Gierlichs, co-promotor

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

June 2016

Analysis and Design of Masking Schemes for Secure Cryptographic Implementations

Oscar Reparaz

Examination committee:
Em. prof. dr. ir. Paul Van Houtte, chairman
Prof. dr. ir. Ingrid Verbauwhede, promotor
Dr. Benedikt Gierlichs, co-promotor
Prof. dr. ir. Bart Preneel
Prof. dr. ir. Patrick Wambacq
Prof. dr. François-Xavier Standaert
(Université Catholique de Louvain, Belgium)
Prof. dr. David Naccache
(École Normale Supérieure, France)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

June 2016

Acknowledgements

Foremost I wish to thank my promotor Ingrid Verbauwhede for offering me a position, her sustained support and always having the office door open for me when it was needed.

I would like to thank my assessors prof. Bart Preneel and prof. Patrick Wambacq for their timely and insightful remarks throughout the doctoral program. I would like to thank the additional members of the jury prof. David Naccache and prof. François-Xavier Standaert for their time and effort invested in this dissertation, and prof. Paul Van Houtte for chairing the jury.

I wish to thank Benedikt for his guidance and time spent with me that helped me to bootstrap. I am lucky to have had the opportunity to write papers together with outstanding colleagues: Josep Balasch, Begül Bilgin, Ruan de Clercq, Thomas de Cnudde, Junfeng Fan, Vincent Grosso, Svetla Nikova, Vladimir Rozic, Sujoy Sinha Roy and Frederik Vercauteren. I learnt a lot from you, and hope to continue writing papers with you.

COSIC was an ideal environment to carry out my research. In one way or another, many people contributed to create a pleasant office environment. I have been lucky to share the 01.62 and B01.24 offices with you, to enjoy together at 13u the Alma cuisine, coffee breaks, Friday beers, summer BBQs, weekends in the Ardennes, football matches and concerts! Péla deserves a very special thanks for her kindness and assistance.

I had a wonderful stay in Leuven. To all my friends I shared rich experiences with: a huge thank you! You know who you are. I found you playing music (jazzing up Mondays), playing squash, tennis, improvising picnics and BBQs, having tough times at oude or doing excursions. Many of you already left Leuven but you know how important you are for me.

I wish to thank the Fund for Scientific Research - Flanders (FWO) for generously funding my research activities, including conference traveling. The interaction with other colleagues at conferences led to many stimulating discussions.

This thesis is dedicated to my parents for their unconditional support and encouragement throughout all these years.

ÓR.
Leuven, spring 2016

Abstract

Masking is the central topic of this thesis based on publications. Masking is a technique that allows the secure execution of cryptographic algorithms in untrusted environments. More concretely, masking provides security guarantees even if an adversary observes side-channel leakage.

We first propose a methodology to attack masked implementations more quickly. Our method is relevant in practice since it allows to carry out attacks that before took months in days. The proposed method first locates the relevant time samples for an attack and then only attacks those. For this purpose we rely on versatile information-theoretic tools.

The second selected paper in this thesis deals with Differential Power Analysis, masking and bit-slicing at very high clock speeds, such as those typically found in today's smartphones and personal electronic devices. We present an attack on an ARM Cortex-A8 running at 1 GHz, and then apply the principles of gate-level masking to develop a DPA-resistant bit-sliced AES implementation.

In our third selected paper, we propose a new masking strategy for a post-quantum public-key algorithm: ring-LWE. Our solution is essentially arithmetic masking with a bespoke probabilistic decoder. Our approach fits in a standard FPGA and incurs manageable performance overheads.

We explain in our fourth paper similarities and differences between theoretical and practical instances of masking schemes. These observations allow us to break some masking schemes proposed in literature and transfer attractive features from one scheme to another.

To conclude, in the fifth paper we describe a simple, yet powerful tool to detect flaws in masking schemes. Sound masking schemes can be surprisingly difficult to design (especially if they provide higher-order security guarantees); our tool assists the design process of a masking scheme by assessing the soundness of a masking scheme at the algorithmic level before implementing it on an actual device.

Samenvatting

Het centraal onderwerp van deze publicatie-gebaseerde thesis is maskering. Maskering is een techniek die het veilig uitvoeren van cryptografische algoritmes in onbetrouwbare omgevingen verzekert. Meer bepaald, masking verzekert beveiliging zelfs wanneer een tegenstander zijkanaal-informatie ontvangt.

Eerst stellen we een methode voor om gemaskeerde implementaties sneller aan te vallen. Aanvallen die eerder maanden lang duurden, nemen nu maar dagen in beslag. De voorgestelde methode zoekt enkel de tijdsmonsters die relevant zijn voor een aanval, en valt enkel die aan. Hiervoor gebruiken we veelzijdige informatie-theoretische instrumenten.

Het tweede geselecteerde artikel in deze thesis gaat over DPA, maskering en "bit-slicing" bij zeer hoge klokfrequenties, zoals typisch gevonden in smartphones en persoonlijke elektronische toestellen. We presenteren een aanval op een ARM Cortex-A8 op 1 GHz, en dan passen we "gate-level" maskeringsprincipes toe om een "bit-sliced" AES implementation te ontwikkelen die bestand is tegen DPA.

In ons derde geselecteerde artikel, introduceren we een nieuwe maskeringsmethode voor ring-LWE, een post-quantum publieke-sleutel algoritme. Onze oplossing is rekenkundige maskering met een op maat gemaakte probabilistische decodeerder. Onze aanpak past in een standaard FPGA en is efficiënt.

In ons vierde artikel leggen we de gelijkenissen en verschillen tussen theoretische en praktische instantiaties van maskeringsschema's. Met deze observaties breken we sommige voorstellen en kunnen we aantrekkelijke eigenschappen van één schema naar een ander overbrengen.

In de laatste paper beschrijven we een simpel, maar krachtig instrument om fouten in maskeringsschema's te detecteren. Goede maskeringsschema's kunnen moeilijk te ontwerpen zijn (vooral als ze hogere-orde beveiliging garanderen); ons instrument helpt het ontwerpproces door de kwaliteit van een maskeringsschema te meten op algoritmisch niveau vóór implementatie op een toestel.

Contents

List of Figures

List of Tables

List of Abbreviations

AES Advanced Encryption Standard

BGW Ben-Or–Goldwasser–Wigderson

BSI Bundesamt fr̈ Sicherheit in der Informationstechnik

CMOS Complementary metal-oxide semiconductor

DES Data Encryption Standard

DH Diffie–Hellman

DPA Differential power analysis

DSA Digital Signature Algorithm

DSS Digital Signature Standard

DUT Device under test

EM Electromagnetic

EMV Europay MasterCard and Visa

FPGA Field-programmable gate array

IDEA International Data Encryption Algorithm

ISW Ishai–Sahai–Wagner

LSB Least significant bit

MAC Message authentication code

MD Merkle–Damgård

NSA National Security Agency

RAM Random-access memory

RNG Random number generator

RSA Rivest–Shamir–Adleman

SHA Secure Hash Algorithm

SPA Simple power analysis

Part I

Analysis and Design of Masking Schemes for Secure Cryptographic Implementations

Chapter 1

Introduction

In this chapter we provide a brief introduction to side-channel attacks. The central topic of this dissertation is masking, which is a countermeasure against side-channel attacks. We omit preliminary concepts from cryptography.

1.1 A brief history of cryptography

Pre-electronic cryptography. Cryptography (Greek for "hidden, secret writing") is mostly about secrets. People have been using codes to prevent unauthorized access to information since the 20th century BC. The techniques have been evolving ever since. Until World War I, cryptography was mainly a manual craft work. Its use was generally restricted to military circles, spies and diplomats.

Contemporary cryptography. After the advent of computers, cryptography saw a significant progress as a scientific discipline and progressively became of civilian use. Today we use cryptography every day to protect our communications by phone, and our electronic transactions for commercial and banking purposes.

Breaking cryptography. Parallel to cryptography is the development of cryptanalysis: the discipline of analyzing and breaking codes. Cryptanalysis comes in two flavors: traditional black-box cryptanalysis and implementation attacks.

Cryptanalysis. Traditional cryptanalysis refers to mathematical breaks. The common denominator of these attacks is that they abstract away from the concrete implementation details. The only information these techniques employ is the

input/output texts from the encryption process. That is, these techniques treat the encryption process as a black box.

Modern cryptography is mature enough so that cryptanalytic attacks against judiciously-chosen cryptographic primitives and protocols are expected to be extremely hard, requiring a significant breakthrough of publicly available cryptanalytical techniques, even when the adversary has access to large computational resources.[1]

Implementation and side-channel attacks. In contrast to purely cryptanalytical techniques, implementation attacks, and more concretely side-channel attacks, consider additional information from the encryption process to break security. Side-channel attacks consider the encryption process as a gray box: the analyst can exploit additional information, such as any physical observable.

Let us use an analogy to explain this difference further. A combination lock can be broken by just trying all possible combinations. This process is expensive. However, if we consider the additional acoustic information obtained with a stethoscope, a skilled locksmith can substantially accelerate the process of guessing the correct combination, by using a divide-and-conquer strategy.

Similarly, by observing the instantaneous power consumption of a modern chip that performs some cryptographic operation, the analyst is able to recover the secrets inside the chip.

The advantage of side-channel attacks is that they often provide a significant speedup compared to a cryptanalytical attack, bringing down the complexity of mounting such attacks to the practical realm. In addition, side-channel attacks are often orthogonal to the intrinsic mathematical strength of the algorithm.

Active vs. passive side-channel attacks. This thesis deals with a class of passive side-channel attacks, namely power analysis attacks. Passive side-channel attacks, in contrast to active side-channel attacks, do not disrupt the natural working procedure of a cryptographical operation. Active side-channel attacks insert faults during the computation and are outside of the scope of this thesis.

References. A detailed treatment of the historical aspects of cryptography is outside of the scope of this thesis. A comprehensive study on the history of cryptography can be found in the monograph of Kahn [Kah96]. Ross Anderson's excellent encyclopedic work [And08] is also of historical interest. An extensive and exhaustive reference for modern cryptography is the Handbook of Applied Cryptography by Menezes, van Oorschot and Vanstone [MvOV96].

[1] This situation will change whenever the adversary has access to a quantum computer, but such a computer has not been publicly built yet. In Section 2.8.2 we elaborate on this.

1.2 A brief history of side-channel attacks.

In this section we give some historical examples of side-channel attacks.

1.2.1 TEMPEST

TEMPEST refers to a series of NSA and NATO standards that regulate electromagnetic emissions and their exploitation to recover intelligible information. This section is based on the following two sources from the NSA. Boak's "A history of U.S. communication security" [Boa73] bundles a series of lectures aimed at interns and other employees from the NSA given in 1966. It was revised in 1973 and declassified in 2008. The 1972 NSA report [Fri72] was written by Jeffrey Friedman and later declassified in 2007.

Initial discovery. The U.S. Army and Navy were using secure teletypewriter communications during WWII. The encryption machine was the Bell-telephone mixing device 131-B2. Friedman claims that in 1943, in the midst of the war, engineers from Bell Telephone were testing the 131-B2 when they realized that each time the machine was actioned, a spike popped up in an oscilloscope in a remote place in the lab. Further explorations revealed that it was possible to recover the plaintext being encrypted by the machine using the remote oscilloscope. The engineers set up a demonstration for military officials to prove that this effect was exploitable in the field. They were placed in a building in Varick Street, downtown New York; about 25 meters across the street was a Signal Corps' cryptocenter. After 1 hour of data capture and 3 hours of analysis, they could recover 75% of the plaintext data that was processed.

Countermeasures. Bell Labs subsequently was appointed to study this phenomenon and possible countermeasures. They identified three different sources and three different countermeasures:

1. Radiation through space and magnetic fields. The protection suggested is shielding.

2. Conducted signals on power or signal lines. The countermeasure hinted is filtering on the lines.

3. Space-radiated or conducted signals. The fix proposed is masking. In this context, "masking" according to Boak is "deliberately creating a lot of ambient electrical noise to override, jam, smear out or otherwise hide the offending signals."

Bell Labs reconditioned the 131-B2 mixer adding shielding and filtering to minimize information leakage over compromising emanations. Effective shielding and filtering proved to be challenging: Friedman notes that the compromising emanations span

over a very large portion of the frequency spectrum, "from DC all the way up to GHz range," and the leaking behavior changed from mixer to mixer. Even the same machinery exhibited different behavior in different enviromental conditions. Thus, the shielding had to be effective against a wide range of signals.

The modifications to the 131-B2 did not come for free: as Friedman tells "the modifications caused problems of heat dissipation, made maintenance extremely difficult, and hampered operations by limiting access to the various controls." Interestingly, none of the countermeasures was actually deployed in the field; instead officers were instructed to prevent covert interception by "controlling a zone about 100 feet in diameter around their communications center."

Friedman also claims that the story was rediscovered by the CIA later in 1951 while inspecting the same device 131-B2.

1.2.2 Compromising emanations

Electromagnetic compromising emanations. For an overview of the historical background of electromagnetic compromising emanations, we refer to the PhD thesis of Markus Kuhn [Kuh03].

van Eck radiation. Although the risk of electromagnetic compromising emanations had been mentioned before (Kuhn traces it back to 1966), Wim van Eck was the first to publicly present a proof-of-concept of eavesdropping a CRT display via electromagnetic compromising emanations in 1985 [vE85]. van Eck found that CRT displays emit EM fields in the UHF frequency range and reconstructed the image shown on the screen. He concludes that this exploitation may be possible for distances up to 1 km. The equipment van Eck uses is very simple and consists of a modified TV. van Eck proposes several solutions to mitigate this issue:

1. decrease the signal level
2. increase the noise level
3. randomize the sequence in which the image is drawn on screen.

It is interesting to note that, broadly speaking, these are essentially the same countermeasures that are used in other fields of emission security, such as side-channel protections.

Kuhn EM compromising emanations. Kuhn presents a more detailed study of EM compromising emanations, describing a different experimental set-up than that of van Eck, and discusses possible countermeasures. Kuhn also extends the work to LCD screens.

Optical compromising emanations. Kuhn reported in 2002 [Kuh02] reading CRT displays at a distance by picking-up the intensity of light emitted by the display, even after diffuse reflections on a wall. The process is not straightforward and requires deconvolution. Kuhn presents experiments reading the content of a screen placed at a meter from a white wall using a PIN photodiode with no direct line-of-sight between them.

Loughry and Umphress showed in 2002 [LU02] that inconspicuous LED status indicators from data communication equipment, including encryption boxes, can emit optical signals that are correlated with the data being handled. The working principle of this phenomenon is very simple: since the LED is normally controlled by the same logic that performs some operation on the data, there is a direct information path flowing from the serial data being handled to the optical output of the LED. The information is typically modulated in optical amplitude, and contrary to electromagnetic compromising emanations, can be picked up with relatively simple equipment. Loughry and Umphress report picking up data from modems, LAN NICs, IP routers, storage devices and others from a distance of 20 m with ease.

1.3 Timing attacks

Side-channel attacks were introduced by Paul C. Kocher at CRYPTO 1996 [Koc96]. The core idea of the paper is that by measuring the time a cryptographic implementation takes to execute, an adversary can recover the secrets involved in the computation. Kocher presents key-recovery attacks on implementations of RSA, DH, DSS and other algorithms.

The preconditions for the attack are the following. Suppose the operation is a modular exponentiation with known base and secret exponent. Assume the modular exponentiation is implemented as a sequence of modular multiplications and squarings. The sequence of operations depends on the secret exponent. In addition, it is reasonable to assume that the modular multiplication takes different time for different inputs.

The attack works as follows. The attacker collects enough measurements of the execution time for the exponentiation. Suppose the implementation handles first the LSB bit of the secret exponent b_0. The attacker can model somehow the execution time based on the known inputs and a guess for b_0. This predictions are contrasted with the measurements, and if they match the guess is deemed correct, and the attack continues in a divide-and-conquer fashion against the next secret bit b_1 carrying over the information learned on b_0.

Historical notes. Bernstein traces back timing attacks to attacks on the TENEX operating system password check routine [Ber04] from the 1970s. This routine compared the user-provided password against the stored password on a character by character basis, aborting the process whenever the two strings differ. Thus, an attacker who observes the time it takes to compare the two strings can infer the point at which both strings differ, and easily use this fact to progressively guess the password.

1.4 Power analysis attacks

Power analysis attacks were introduced by Paul C. Kocher, Joshua M. Jaffe and Benjamin C. Jun in a technical report in 1998 [KJJ98c] and later at the CRYPTO 1999 conference [KJJ99]. This is a landmark paper, introducing a whole new area of research and having a significant impact on the industry.

Power analysis inspects the instantaneous power consumption to find cryptographic keys from embedded devices. Contrary to other more invasive approaches, power analysis can be easy to implement if no countermeasures are present, does not require expensive equipment and the cost per device is low, that is, the attack scales economically.

1.4.1 Acquisition of power traces

It is hard to give a general methodology for acquiring power traces. Each device has its own particularities. We describe here the basic method.

Canonical method. A traditional method is to insert a shunt resistor in the ground path and measure with a digital oscilloscope the voltage drop over the resistor. This voltage is related to the instantaneous power consumed by the device under test (DUT). The resistor value is typically low enough to prevent DUT under-voltage.

A triggering mechanism is needed to start capturing while the device is performing the cryptographic operation. The sampling frequency depends on many factors, including the DUT running frequency. The sampling frequency is normally in the range of hundreds of MS/s to a few GS/s.

Instantaneous power or EM. There are many magnitudes that are related to the instantaneous power consumption, and that can be used instead to perform power analysis attacks. The most common is the electromagnetic (EM) radiation. There are many points to pick up EM radiation suitable for power analysis: either in the

vicinity of the silicon die that is performing the crypto to any component in the power supply line (such as decoupling capacitors). The advantage of EM analysis is that it requires almost no modifications to the DUT; the disadvantage is that it is typically more complex to carry out since more variables, such as the probe location and orientation, should be taken into consideration.

1.4.2 Simple power analysis

Simple power analysis (SPA) encompasses a wide range of analysis techniques. They typically analyze one or a few traces, and inspect patterns therein. If successful, SPA can completely break a cryptographic implementation using very few traces.

SPA attacks are highly dependent on the exact details of the implementation. Hence, they are often everything but simple, despite their name. The canonical examples are mainly in the domain of public-key implementations but secret-key implementations can also be vulnerable.

A naive example. Say a device is performing an RSA signing operation. This operation essentially raises a public input m to a private exponent d to produce the signature $m^d \pmod N$. If the exponentiation uses a naive square-and-multiply algorithm, then the sequence of performed square and multiply operations reveals the secret exponent d. This sequence of operations may be distinguishable in the power trace (say they are different code routines), and thus may be susceptible to a SPA attack.

1.4.3 Differential power analysis

Differential power analysis (DPA) typically analyzes up to hundreds or millions of traces to recover cryptographic secrets. DPA exploits the fact that the power consumption is correlated with the data value being handled by the device.

DPA works by first modeling the power consumption when the device is handling a concrete intermediate value. This modeling involves a guess of a key portion. Then, the different key-dependent models are compared against measurements. The key guess that leads to a better fit is deemed as the correct key candidate.

DPA has many attractive properties: it is resilient to noise (the success rate can be amplified by using more traces) and does not require extensive computational resources.

There are many variations on SPA and DPA. A very popular variant is Correlation Power Analysis [BCO04]. One can consider profiled variants where the adversary can perform a prior profiling with a study device under his control. These are called

Template Attacks and were introduced at CHES 2002 by Suresh Chari, Josyula R. Rao and Pankaj Rohatgi [CRR02]. Another variant is collision attacks. They detect internal collisions during the executions to recover key material, and the idea is due to Hans Dobbertin as Kai Schramm, Thomas Wollinger and Christof Paar explained at FSE 2013 [SWP03].

1.4.4 A peek at the leakage causes

In this thesis we abstract away from the causes and work assuming that the power consumption is somehow correlated with the data being handled by the device. (Often, we do not even know what the precise relation is.) However, it is useful to present the root causes of the data-dependent power consumption in static CMOS, a very common logic style.

Three contributions. There are three main components in the power consumption of CMOS circuits [Rab96]. They can be categorized into either static or dynamic power consumption.

The static power consumption in CMOS is very low. It is caused by the leakage current of transistors. Traditionally it has been neglected for side-channel analysis purposes, but for forthcoming technologies it may bring interesting results, since static power consumption increases as technology scales down.

The dynamic power consumption is more often exploited in side-channel analysis. There are two components of dynamic power consumption: due to switching current and short-circuit current.

- Switching current is due to the process of charging and discharging the capacitance of internal nodes of a CMOS circuit. This current depends on the internal parasitic node capacitance (mostly gate and wire capacitance). Long buses, for example, exhibit large capacitance, requiring large amounts of current to charge or discharge.
- Short-circuit current is due to a current path from power line to ground while both nMOS and pMOS transistors are on. This only occurs for a very short time and can be minimized if rise and fall times are kept low.

Thus, dynamic power consumption depends on the data handled in the circuit.

1.5 Masking

Masking is a countermeasure against power analysis attacks. Masking decouples the processed values by the device from the intermediate values of a cryptographic

algorithm. It does so by probabilistically splitting each bit of the original computation into several shares, so that each of the shares considered individually does not yield any information on the original, native bit. A crucial aspect is that computation is carried out by handling shares only, without reconstructing the original, native bit.

The IBM research proposal. At the same conference where SPA and DPA was introduced, a team from the IBM Thomas J. Watson research center published a "sound countermeasure" against power analysis attacks, namely masking [CJRR99]. Suresh Chari, Charanjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi propose a masked encoding of each original bit b into k shares

$$b \oplus r_1, r_2, \ldots, r_{k-1}, r_1 \oplus \cdots \oplus r_{k-1} \tag{1.1}$$

where each r_i is a random iid bit.

Furthermore, they provide a formal security proof arguing that this probabilistic encoding makes power analysis attacks more difficult in terms of traces needed. More precisely, the main result is that an adversary willing to distinguish between two leakage distributions of masked bits $b = 0$ and $b = 1$ needs exponentially more traces as σ^{2k}, where σ is the noise level and k the masking order. This proof is given in the noisy leakage model (cf. page 45) and is valid asymptotically.

The duplication method. A similar concept of masking was introduced at CHES 1999 by Goubin and Patarin with the name "the duplication method" [GP99]. Hence, masking is often credited to both the IBM team and Goubin and Patarin. Goubin and Patarin give practical constructions for computing DES and RSA. For DES they give essentially a table-based approach to compute the non-linear functions, and provide some optimizations to reduce RAM usage. (The optimizations were found later to be slightly flawed, see for example the 2007 article of Fumaroli, Mayer and Dubois [FMD07].)

1.6 Other countermeasures against side-channel analysis

Filters. An obvious approach to mitigate power analysis attacks is to attenuate the fluctuations in the power consumption by introducing analog filters in the power line of the device. This requires analog circuitry, and normally attenuates, but does not completely eliminate, exploitable leakage. Hence, it is rarely used as the sole countermeasure. Note that this countermeasure is easily circumvented by EM analysis. A related, ingenuous approach based on capacitors has been patented by Shamir [Sha99].

Adding noise. Another natural countermeasure to side-channel analysis is to add noise to the side-channel signal so that it becomes harder to exploit. In the case of power analysis, this can be done by running in parallel to the cryptographic operations some dummy process that inserts noise in the power traces. The amount of required noise has to be carefully calculated. Oftentimes, this countermeasure is not implemented alone, but along with some other countermeasure that benefits from noise (for example, masking.)

Special logic styles. It is appealing to try to solve the side-channel problem at the circuit level. One approach is to use specially designed logic styles with (in theory) data-independent power consumption. Examples of logic styles that aim at data-independent power consumption are dynamic and differential logic, such as Sense Amplifier Based Logic (SABL) [TAV02a] or Wave Dynamic Differential Logic (WDDL) [TV04a]. Differential logic means that the signal is encoded in a pair of wires; dynamic logic evaluates a combinatorial circuit in two phases: precharge and evaluation. The careful combination of both can lead to (theoretically) gates with data-independent power consumption.

However, those logic styles often rely on assumptions (for example: symmetry, or signal propagation characteristics) that are not fully met in practice. Those imperfections may render these approaches vulnerable [SS06].

Protocol level. Another kind of countermeasure against side-channel analysis is to limit the exposure of a cryptographic key, by rotating and changing it often enough. The number of executions of the cryptographic implementation under a given key is then limited by a fixed counter.

The typical setting is to periodically derive a session key $k_i = \mathrm{KDF}(k, i)$ from a master key k at epoch i. The session key k_i is only used for a handful of times, after that the session key is disposed of and another one is derived. The adversary is then limited by the number of available measurements under the same key. Note that the key derivation function KDF should also be protected, and that transmitter and receiver have to be synchronized.

1.7 Outlook for the rest of this thesis

The rest of this thesis deals exclusively with masking. In Chapter 2 we provide an introduction to the design of masking schemes. In Chapter 3 we provide an introduction to their analysis.

The scientific contribution of this thesis is based on publications. I selected five representative publications to include in this dissertation:

- Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings, volume 7428 of Lecture Notes in Computer Science*, pages 155—174. Springer, 2012.
- Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1 GHz. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2015.
- Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 683—702, 2015.
- Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764—783. Springer, 2015.
- Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In Thomas Peyrin, editor, *Fast Software Encryption, 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Lecture Notes in Computer Science*, 20 pages. Springer, 2016.

These five papers deal with *particular* masking schemes tailored toward specific cryptographic algorithms as well as with *generic* masking schemes that could be applied more generally to a wider class of cryptographic algorithms. In these selected five papers we address practical *design* issues as well as the *analysis* of masking schemes. A map of the representative publications included in this thesis can be found in Figure 1.1, along the mentioned two criteria.

The complete list of publications can be found on page 67.

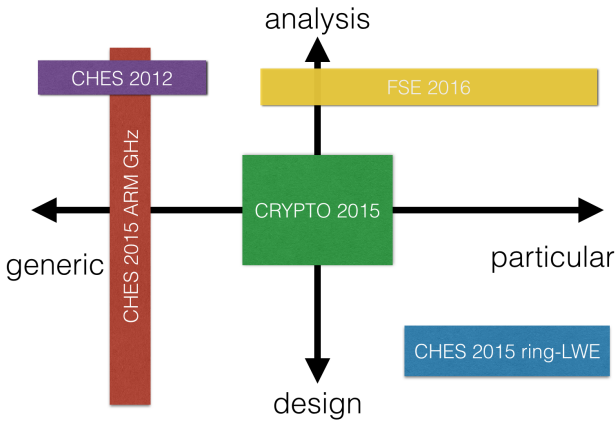


Figure 1.1 – Map of representative publications included in this thesis.

Chapter 2

Design of masking schemes

This chapter intends to provide a general overview of the various masking schemes present in the literature. We first give a general definition of a masking scheme in Section 2.1, and then present first-order techniques in Sections 2.4 and 2.5. We finally describe higher-order techniques in Section 2.6.

Our contributions to the design of masking schemes are described in Section 2.8.

2.1 What is a masking scheme?

A masking scheme is a specific countermeasure designed to make side-channel attacks more difficult to carry out. A masking scheme defines a *data representation* mechanism and a *computation* procedure to operate on masked data.

Data representation. The masked data representation is a reversible probabilistic mapping $x \mapsto m(x)$ used to represent a data value (string of bits) x. The mapping is such that if an adversary gets a noisy version of the masked representation $m(x) + \eta$, it is difficult for him to recover the underlying native value x.

A common mapping is Boolean masking. Boolean masking splits a variable[1] $x \in \mathrm{GF}(2)$ into a set of shares $(x_1, \ldots, x_n) \in \mathrm{GF}(2)^n$ such that

1. the shares sum to x, $\sum x_i = x$; and
2. no proper subset of $\{x_1, \ldots, x_n\}$ gives any information on x.

[1] To be mathematically precise, the following description should be in terms of random variables. However, for simplicity we adopt a more informal language, and abuse notation to denote with the same term a random variable and its realization.

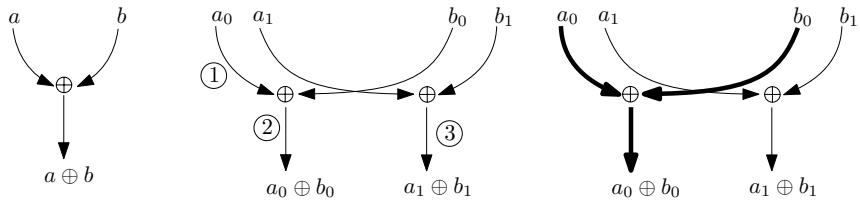


Figure 2.1 – Left: an unmasked circuit. Center: a masked circuit. Right: an unbalanced masked circuit.

Boolean masking can be extended to mask bit vectors (words) by applying the same principle component-wise. Another common mapping is arithmetic masking. Arithmetic masking splits a k-bit variable $x \in \mathrm{GF}(2^k)$ into a set of shares $(x_1, \ldots, x_n) \in \mathrm{GF}(2^k)^n$ such that the shares sum to x, and no proper subset of shares provides information on x.

Data computation. The computation procedure F is an algorithm description to compute a specific function f in the masked domain, that is, accepting and returning data in the m-representation. An important requirement on F is that this m-representation should be preserved throughout the whole computation. In other words, all intermediates appearing in the computation of F should also be represented in the m-domain.

2.2 A simple example

Let us explain the basic concept with a simple example. This toy example allows us to exhibit some important properties of masking.

Perhaps the simplest masked data representation mechanism is first-order Boolean masking. We probabilistically split each native bit b into two bit shares $m(b) = (b_0, b_1)$ such that the shares sum to $b_0 \oplus b_1 = b$ and each share b_i is uniformly distributed in $\{0, 1\}$. To perform this initial sharing, a source of randomness is required. This masking satisfies the two properties of Boolean masking laid out above. Observe that even if we give out all information on (only) one share b_i, the adversary learns nothing about the underlying native bit b.

Consider a simple masked computation procedure. Suppose we want to compute the xor between two bits $f(a, b) = a \oplus b$. This computation is represented in Figure 2.1, left. Assume that the input bits a and b are secret. The masked computation procedure could be something like $F(m(a), m(b)) = (a_0 \oplus b_0, a_1 \oplus b_1)$. This is represented in Figure 2.1, middle. This construction clearly computes the correct result, since $m(a) \oplus m(b) = a_0 \oplus b_0 \oplus a_1 \oplus b_1 = a \oplus b$.

We will see now why this splitting increases the resistance against side-channel attacks in different scenarios of increasing complexity.

Security of individual nodes. Suppose for the moment that the adversary can sniff on a single node and recover its average value (for example, via averaging the side-channel signal), but does not learn anything from a single observation. The goal of the adversary is to recover the underlying native, unmasked value. The unmasked input stays constant across measurements, but the masks are fresh for each measurement.

It is easy to see that every intermediate variable occurring in the masked computation is indeed in the masked representation. The intermediates are of the form a_i, b_i or $a_i \oplus b_i$. The expected average power consumption for each intermediate variable is constant for each unmasked input value, and thus does not reveal anything about the secret inputs.

For example, let us focus on the node (2) in Figure 2.1, middle. The expected value of the value $a_1 \oplus b_1$ is 0.5, irrespectively of the value of a or b. Hence, sniffing $a_1 \oplus b_1$ gives no information on the inputs.

(More precisely, when the adversary probes $a_1 \oplus b_1$, he actually gets $L(a_1 \oplus b_1)$ where L is a (potentially unknown) leakage behavior function that captures the leaking effect of the device. Note that as long as L is linear, the security results hold.)

Superposition of leakage. It is a bit unrealistic to assume that the adversary can record the power consumption of single wires. (Very localized EM measurements could achieve this, but standard power measurements cannot.) Typical measurements contain the superposition of contributions of individual nodes.

Suppose that the adversary sees the (additive) superposition of several nodes. (Say node (2) and (3).) Still, the average signal that the adversary observes does not depend on any secret value. The average signal of each contribution does not depend on any secret value, and hence the average signal of the superposition neither depends on any secret value.

Unbalanced nodes. Suppose now that the circuit is built in a technology with little control over the circuit symmetry. Suppose that the power consumption of a certain branch is dominant, as Figure 2.1, right, shows. It is easy to see that the observations of the previous paragraph also apply. Namely, the average power consumption of this circuit will not reveal anything about any underlying value, even if the superposition is "unbalanced."

Assumptions on leakage behavior. In this example we can already spot an important property of masking. Masking relies on a few assumptions on the leakage behavior. Unlike other approaches, such as differential logic styles, masking does not require that each node consumes the same amount of energy (balanced power consumption). This is highly relevant in practice, since it is difficult to design circuits (ASICs or FPGAs) with balanced power consumption. Masking requires that there is no interaction between different nodes nor second-order terms in the power consumption.

Assumptions on the RNG. Masking requires fresh random numbers for each execution. Moreover, this randomness is secret for the adversary. If it is possible to predict somehow the RNG output, masking provides no security.

Where masking breaks: joint leakage, cross-talk. It is also interesting to see where masking breaks. Suppose that there is a cross-talk component in the power consumption between node (2) and (3). That is, the power consumption contains a term proportional to the product of the individual contributions (2) and (3). The average of the cross-talk term $L(a_1 \oplus b_1) \cdot L(a_2 \oplus b_2)$ depends (in the general case) on the value of the unmasked output bit $c = a \oplus b$, and hence can be used to recover the secret value c. Similar observations apply if there are second-order terms for example in node (1).

Note that the second-order effect can be created by the circuit or can be enforced by the adversary by means of a suitable processing of the measured traces. That is, the leakage behavior L can be enforced to be actually $L_1 \circ L_2$, where L_1 is a non-linear post-processing applied by the adversary. This is the essence of higher-order attacks.

2.3 Construction of linear operations

Designing a masked computation procedure is, in general, not trivial. In the next two sections, we will describe masked computation procedures.

The easiest operations to mask are linear ones. We can generalize the construction of the previous example. To perform a linear operation $y = L(x)$ (with respect to the group operation used to mask the underlying variable x) on masked data x_i, it suffices to apply the operation individually to each share, $y_i = L(x_i)$. It is obvious that the result is correct, that is, $\sum y_i = L(\sum x_i)$. This follows straight from the definition of linearity.

In Boolean masking, this means that the xor operation is efficient and easy to mask. This operation is ubiquitous in secret-key algorithms. For example, in AES all

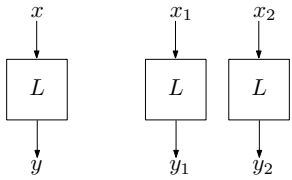


Figure 2.2 – Left: the linear operation to be masked. The input x is split into two shares x_1 and x_2. The output y is also split into two shares y_1, y_2. If the function L is linear, the output can be computed as simply $y_i = L(x_i)$.

the operations are GF(2)-linear except for the pseudo-inversion in GF(2^8) inside SubBytes.

Arithmetic masking is compatible with modular addition. There are a few secret-key algorithms that rely on modular arithmetic for some parts (IDEA and MAC constructions based on MDx or SHA among others); arithmetic masking is thus efficient on those parts. There are plenty of public-key algorithms as well that rely on modular arithmetic.

The hard task is to mask non-linear operations. There are many constructions that we describe in the sequel.

2.4 Generic first-order masking constructions

In this section we describe different first-order masking constructions for non-linear functions. The constructions in this section are generic in the sense that they can be applied to mask, in principle, any Boolean function.

Definition of first-order masking. We define "first-order masking" as a masking scheme that is designed to withstand first-order DPA attacks. First-order DPA attacks are defined in Section 3.1.1 as attacks exploiting information residing in the mean power consumption.

2.4.1 Masked tables

Paul C. Kocher, Joshua M. Jaffe and Benjamin C. Jun from Cryptography Research filed a patent in June 1998 [KJJ98a,KJJ98b] already hinting at masked tables. The IBM research team also suggests masked tables [CJRR99] in 1999, and Thomas S. Messerges reports masked implementation of the AES finalists using masked tables at FSE 2000 [Mes00a] for a 32-bit ARM processor.

The principle is quite natural. It is suitable mainly for software implementations and can result in reasonably fast code. Suppose we want to perform the Sbox lookup S. We first pre-compute a table S' as

$$S'[x] := S[x \oplus m_{\text{in}}] \oplus m_{\text{out}}. \tag{2.1}$$

This table S' accepts inputs that are masked with the mask m_{in} and outputs values masked with the mask m_{out}. By construction S' "unmasks" the input, performs the table S lookup and "masks" the output; all the three steps happen in a single shot without producing side-channel leakage of unmasked intermediates.[2]

Note that the table S' is tied to a specific value of the input and output masks $(m_{\text{in}}, m_{\text{out}})$, and thus S' has to be recomputed each time new masks are drawn. Ideally, this happens at the beginning of each execution. This re-computation of the table S' incurs a performance overhead. Note also that in a typical smart-card the table S' has to be stored in RAM memory since its entries change over time. This limits the size of the table S.

2.4.2 Gate-level masking

The idea of gate-level masking is to mask each logic gate present in the circuit. The main advantage is that it decouples the design of the digital circuit from the issue of providing resistance against side-channel attacks. In other words, the problem is (in theory) reduced to masking a small set of logic gates. These masked gates can be freely composed to build any logic circuit.

Trichina gate. Trichina proposes in [Tri03] a masked AND gate. This is one of the earliest proposals of gate-level masking. This idea also appears in a paper by Golić and Menicocci [GM04].

The basic idea to design the masked AND gate is to apply the distributive property of multiplication over addition to the unmasked output bit $c = ab$. Say a and b are input bits shared into (a_0, a_1) and (b_0, b_1), and c is the output bit. Then

$$ab = (a_0 + a_1) \cdot (b_0 + b_1) \tag{2.2}$$
$$= a_0b_0 + a_0b_1 + a_1b_0 + a_1b_1. \tag{2.3}$$

(Here variables are bits, product is AND and addition is XOR.) Trichina notes that if the masked output bit is a_0b_0, then the correction term (other share) is $a_0b_1 + a_1b_0 + a_1b_1$. She even proposes to use an extra fresh random bit r as the output mask. This small modification, although not explicitly noted in the

[2] This already suggests an ideal side-channel countermeasure: implementing the whole computation in a single look-up table. However, the size of the table is normally prohibitive.

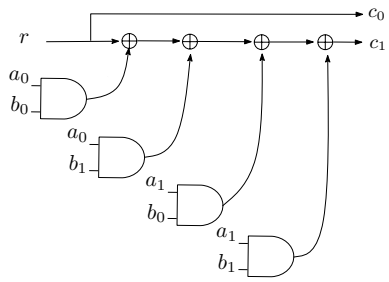


Figure 2.3 – Masked AND as per Trichina.

manuscript, allows to compose these masked gates. She proposes one output share c_0 to be the fresh random bit and accordingly modify the second output share c_1:

$$c_0 = r \tag{2.4}$$

$$c_1 = (a_0 b_0 + r) + a_0 b_1 + a_1 b_0 + a_1 b_1. \tag{2.5}$$

This computation is represented in Figure 2.3. Please note the importance of calculating the individual operations in the correct sequential order. This is unusual in conventional design tools and requires manual intervention by the digital designer.

Infineon masked gates. Franz Klug, Oliver Kniffler and Berndt M. Gammel from Infineon patented [KKG02] in early 2002 several AND, OR, NOR, NAND and MUX masked gates. They compose those masked gates to construct a full adder. A remarkable feature of their solution is that all inputs to masked gates share a common mask. The output is as well protected by the same mask. This opens the possibility of using a single 1-bit mask for the whole circuit, reducing the requirements in area and on the randomness needed (albeit making second-order attacks easier.)

Motorola masked MUX. Thomas S. Messerges, Ezzat A. Dabbish and Larry Puhl from Motorola filed in July 1999 a patent [Tho01] for a generic masking method that can be applied at the gate level. The structure and working principle of this method is fairly simple and elegant.

The core of the method is a masked MUX description with a masked selection signal. A masked MUX is described in Figure 2.4. For the implementation of this masked MUX, they use cross-bar switches[3] and (unmasked) MUXes.

[3] A 2-bit cross-bar switch takes 2 input bits, one control bit and outputs two bits. If the control bit is off, the output is simply a copy of the input; if the control bit is active the output is the swapped version of the input bits.

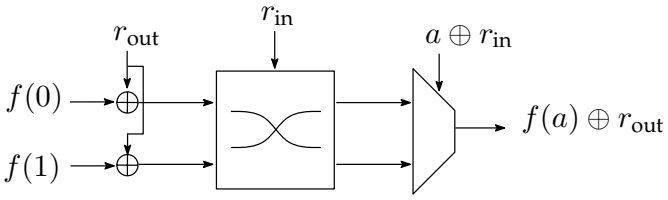


Figure 2.4 – Masked MUX as per Motorola.

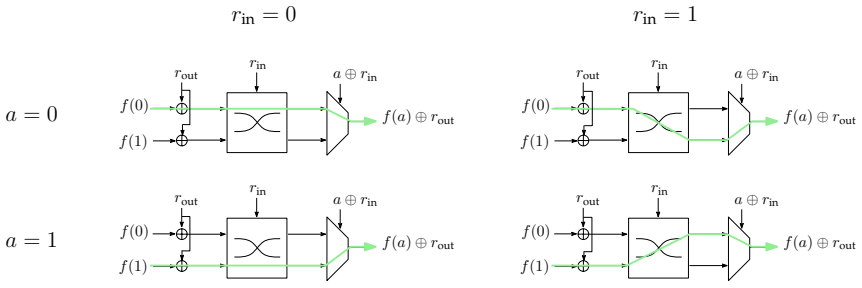


Figure 2.5 – The four possible signal paths in the Motorola MUX construction.

The selection signal of the MUX is randomized with a mask. To ensure the correctness of the output, the input to the MUX is "corrected" with the cross-bar switch. The selection signal of the cross-bar switch is randomized as well. A nice feature is that the input and output masks are decoupled, and thus the output mask does not interfere in the computation. The four possible signal paths in this construction are depicted in Figure 2.5.

From here, one can mask any Boolean function by decomposing it into 2-to-1 MUX operations by recursively applying Shannon's decomposition [Sha49]. The expected circuit size is, however, quite large. The authors present applications to DES.

Infineon glitch-aware masked gates. Glitches occurring in standard CMOS may render masking insecure. This is explained in Section 3.3.1 (page 42). Wieland Fischer and Berndt M. Gammel study at CHES 2005 [FG05] gate-level masking in the presence of glitches. In a theoretical study, they show that there is no set of universal masked gates with 2 input bits and 1 output bit split into 2 shares that resist glitches in their model.

On the further assumption that the gate delay is equalized, and the output capacitances are equal, they do give constructions for masked AND and OR gates. The construction is very similar to Trichina's; the interesting part here is the theoretical treatment and modeling of glitches.

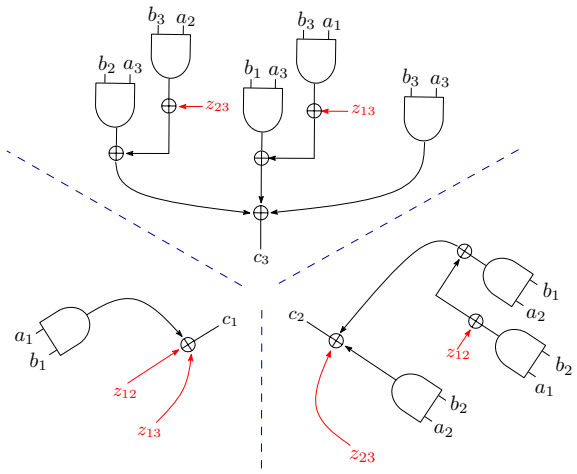


Figure 2.6 – ISW secure AND (secure against single probe).

Ishai—Sahai—Wagner private circuits. Ishai, Sahai and Wagner published at CRYPTO 2003 [ISW03] (the "private circuits paper") a generic approach to secure computation in the presence of a probing adversary. Their main contribution is a secure AND gate at any order, along with a convenient framework to prove its security. The secure gates ("gadgets" in their terminology) can be freely composed and are built from ideal logic gates.

The basic idea is the same as the Trichina gate: if we want to compute the product $c = ab$ from the shares a_i and b_i, we first compute all cross-products $a_i b_i$ and then carefully xor them along with some fresh randomness. The trick here is to very carefully select which cross-products to xor together, and in which order to add the randomness. A smart choice allows to build an elegant security proof.

In Figure 2.6 we draw the circuit secure against adversaries using a single probe. The general construction is described in pseudo-code in Figure 2.7. In the description, brackets are important and show the order that the computation should follow. An intermediate state of the computation is shown in Figure 2.8.

The main contribution of the ISW paper is the simulation-based proof of security. This proof is given in the probing model. (This model is treated in Section 3.5.1.) A very powerful property of the proof is that it ensures that the composition of gadgets is still secure.

While theoretically sound, private circuits can lead to insecure implementation if implemented in hardware that glitches. Nevertheless, ISW strongly influenced other higher-order masking schemes. This is discussed in Section 2.6.2.

Require: s-shares $\mathbf{a}$ and $\mathbf{b}$
Ensure: s-shares $\mathbf{c}$ satisfying $c = ab$
  for i from 1 to s **do**
    for j from $i+1$ to s **do**
      $z_{ij} \leftarrow \texttt{rnd()}$
      $z_{ji} \leftarrow (z_{ij} \oplus a_i b_j) \oplus a_j b_i$
    end for
  end for
  for i from 1 to s **do**
    $c_i \leftarrow a_i b_i$
    for j from 1 to s, $j \neq i$ **do**
      $c_i \leftarrow c_i \oplus z_{ij}$
    end for
  end for

Figure 2.7 – ISW algorithm in pseudo-code for an AND gate.

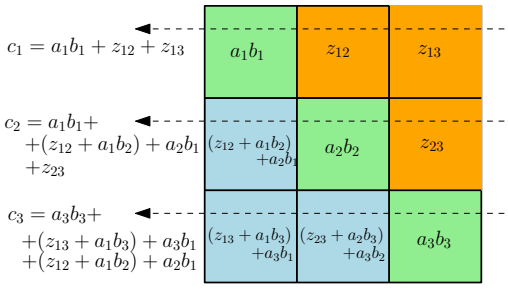


Figure 2.8 – Intermediate state of the ISW computation for $s = 3$.

2.4.3 Threshold implementations

Threshold Implementations were introduced by Svetla Nikova, Christian Rechberger and Vincent Rijmen in 2006 [NRR06]. (See also the journal publication [NRS11].) For an extensive treatment of the subject, we refer to the PhD thesis of Begül Bilgin [Bil15].

Central idea. The central idea of threshold implementations is the concept of non-completeness. A combinational circuit is non-complete if it is possible to partition it in a way such that each subcircuit does not see all input shares of the original circuit.

Resistance against glitches. The non-completeness property is stronger than just the "properly masked" property (= all intermediate variables masked.) This property implies protection against DPA even if the combinational blocks glitch (cf. Section 3.3.1). This guarantee applies irrespectively of the exact glitching behavior.

Security argument. An elegant feature of threshold implementations is its simple security argument to provide security in the presence of glitches. Glitches originate from a combinational block. Each combinational block does not see all input shares, hence each combinational block knows nothing about the (secret) underlying native

(unshared) value. Therefore, glitches cannot provide any information on the native value.

Assumptions. There are standard assumptions in place. Mainly, each sub-circuit leaks independently of others (in particular, this means there are no second-order effects such as crosstalk). These assumptions are standard in masking.

From combinational circuits to sequential logic. In practice, almost all cryptographic algorithms are implemented with memory elements. A threshold implementation ensures that each combinational stage is non-complete. The composition of several combinational stages (with registers in between) needs not to be non-complete, as the registers serve as a synchronization mechanism and do not further propagate glitches. However, an important precondition is that each stage sees a uniform sharing at the input. This can be achieved by remasking, or can be ensured by the previous stage. In this case, the previous stage is called a uniform sharing.

Uniformity. A requirement of threshold implementations is that the input shares have full entropy (this is called "uniform sharing" in some publications). If the output shares of a combinational stage have also full entropy, then the output can be directly plugged into the next combinational stage. This is convenient since it avoids a refreshing. However, not all non-complete sharings satisfy this property (preserving full entropy input/output). The sharings that satisfy this property are called uniform.

Example: synthesis of an AND gate. We illustrate here the design process of a threshold implementation of the 2AND function. Suppose we start with three input shares and three output shares. The native function $c = f(a, b)$ we want to share is $c = a \cdot b$.

The input bit a is shared into three bits a_1, a_2, a_3 such that $\sum a_i = a$ (respectively b.) The output bit c will also be shared into three bits c_1, c_2 and c_3 such that $\sum c_i = c$. The problem is to define three component functions f_1, f_2 and f_3 that compute c_1, c_2 and c_3 such that non-completeness is preserved.

The sum of the output shares is easy to express as function of the input shares. We have that

$$\begin{aligned} c_1 + c_2 + c_3 = {} & a_1b_1 + a_1b_2 + a_1b_3 + && (2.6)\\ & + a_2b_1 + a_2b_2 + a_2b_3 + && (2.7)\\ & + a_3b_1 + a_3b_2 + a_3b_3. && (2.8) \end{aligned}$$

This is enforced by the construction of f, and by the fact that the output should be correct. Now the problem is to distribute the right hand side terms into different buckets for c_1, c_2 and c_3, such that non-completeness is preserved. For example:

$$c_1 = a_2b_2 + a_2b_3 + a_3b_2 \tag{2.9}$$
$$c_2 = a_1b_3 + a_3b_1 + a_3b_3 \tag{2.10}$$
$$c_3 = a_1b_1 + a_1b_2 + a_2b_1. \tag{2.11}$$

We note that $c_i = f_i(a_{i+1}, b_{i+1}, a_{i+2}, b_{i+2})$ does not depend on a_i nor b_i. The reader can extend this reasoning to other f functions and other numbers of input or output shares. The trick is to express the algebraic normal form of f as a function of the input shares, and to distribute accordingly each term into the output shares.

Implementations. There are many threshold implementations proposed. We list here a few: Noekeon [NRS09], Keccak [BDPV10], Present [PMK+11], AES [MPL+11], and all 3×3 and 4×4 S-boxes [BNN+12].

2.5 Particular masking constructions

There are masking schemes specifically tailored towards concrete cryptographic algorithms. They typically benefit from specificities of the algorithm to achieve a better performance when compared to generic methods.

We can classify the specific masking constructions into two large groups: those suitable for secret-key algorithms and those more adequate for public-key algorithms. Two prominent secret-key algorithms are the AES and DES block ciphers. Masking, when applied to public-key algorithms is often referred to as blinding. Blinding is a related concept to masking, and refers to "the use of random, but invertible, transformations that are neutral to the cryptographic operation." [KLL+11]

2.5.1 Particular constructions for AES

Rijndael is the de facto universal standard block cipher. Rijndael was first published in 1998 and selected as the US AES in 2001 after a public selection process by NIST. Rijndael is designed by Vincent Rijmen and Joan Daemen.

The only non-linear component of the AES is the Sbox. It is based on the finite field (pseudo-)inversion[4] $x \mapsto x^{-1}$; this construction is due to Nyberg [Nyb93].

[4] This pseudo-inversion maps 0 to 0.

Software. There are several proposals that exploit the algebraic structure of the Sbox. The problem is to compute the masked inversion $(x^{-1} + m_2, m_2)$ from a masked value (x, m_1). Mehdi-Laurent Akkar and Christophe Giraud propose in CHES 2001 [AG01] to switch between Boolean and multiplicative masks. This multiplicative masking is compatible with inversion, after the inversion the masking is reverted to Boolean. This countermeasure is efficient in software. However, it is vulnerable to first-order DPA since it does not mask the zero value, as shown by several sources (Golić and Tymen [GT02] at CHES 2002; Akkar and Goubin [AG03]; Trichina, De Seta and Germani [TSG02] at CHES 2002.) There are approaches to circumvent this.[5]

Hardware. In hardware there are many alternatives. Most of them rely on subfield arithmetic to perform the inversion in $\mathrm{GF}(2^8)$. The basic idea is to compute the inverse in $\mathrm{GF}(2^{2k})$ with arithmetic in $\mathrm{GF}(2^k)$, by representing $\mathrm{GF}(2^{2k})$ as an extension field of $\mathrm{GF}(2^k)$. The problem is then reduced to masking operations in smaller fields. (This idea can be applied as well to design small area unprotected Sboxes.) See the work of Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller and Vincent Rijmen [OMP04, OMPR05] or the work of Elena Trichina [Tri03, TKL04].

2.5.2 Particular constructions for DES

DES and its variants such as triple DES are popular legacy block ciphers. They are still widely used in the banking sector (see for example the 2006 specifications of the EMV payment standard [Con08]); yet, triple DES is typically slower and provides less security than AES. It was designed by a team at IBM (and subsequently modified by the NSA) in the early 1970's.

The Sboxes of DES do not feature any (public) structure. Hence, one typically resorts either to generic methods or algebraic interpolation.

Sboxes as polynomial functions. The idea here is to express the Sbox as a polynomial function $x \mapsto \sum a_i x^i$ over a finite field, where a_i is a set of coefficients (found via interpolation) fixed for a given Sbox. This evaluation can be performed with additions, multiplications with constant, squarings and multiplications. The difficult operation is the multiplication, and one typically resorts to ISW-like schemes. This principle can be applied to any Sbox. For the special case of the DES Sboxes, see the series of papers by Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, Matthieu Rivain at FSE 2012 [CGP+12]; the improvement of Arnab Roy and Srinivas Vivek at CHES 2013 [RV13]; and the follow-up paper

[5] The idea is to detect when the input is 0, and react accordingly, without leaking. Similar observations have been made in the multi-party computation literature [DK10].

by Jean-Sébastien Coron, Arnab Roy and Srinivas Vivek at CHES 2014. [CRV14, CRV15]

2.5.3 Particular constructions for RSA and DH

For systems based on modular exponentiation, such as RSA and Diffie-Hellman, a solution was already hinted by Kocher in 1996 [Koc96] using the principle behind David Chaum's 1982 blind signatures [Cha82]. (One might think of blind signatures as tangentially related to Shamir's no-key protocol for key exchange, aka Shamir's three-pass protocol [MvOV96, p. 500], also originating in the early 1980's.)

Basis blinding. The basic idea of basis blinding goes as follows. Suppose we would like to protect the exponentiation $g^x \pmod p$ where the basis g and the modulus p are public and the exponent x is secret. The core idea is to randomize the input g to the exponentiation, such that the adversary can no longer make hypotheses on intermediates of the exponentiation. A random, secret and ephemeral value r is generated prior to the exponentiation and the value $(r^{-1})^x \pmod p$ is computed. The input g is multiplied by the random value r and then the product is exponentiated $(r \cdot g)^x \pmod p$. Note that in this exponentiation the basis $r \cdot g$ is unknown to the adversary, and hence predictions on intermediate variables are hard to make. Finally, the output $(r \cdot g)^x \pmod p$ from the exponentiation is corrected by multiplying by $(r^{-1})^x \pmod p$ to yield the intended result $(r^{-1})^x \cdot (r \cdot g)^x \pmod p = g^x \pmod p$.

Other blindings. Due to the industrial relevance of the RSA, Diffie-Hellman and DSA algorithms, there are many variants and optimizations of this basic idea. In addition, the mathematical structure of these algorithms allows efficient blinding countermeasures. The previous idea can be extended to also randomize the exponent or the modulus. A good overview can be found in the 2013 BSI publication by Dirk Feldhusen, Guntram Wicke, Arnold Abromeit and Lex Schoonen [FWA+13].

2.5.4 Particular constructions for ECC

Not surprisingly, due to the rich mathematical structure of elliptic curves, there is ample room for randomized computations. In CHES 1999 [Cor99] Coron already proposes three different randomization mechanisms: scalar blinding, base point blinding and Z-coordinate randomization.

Scalar blinding. Scalar blinding consists of adding a random multiple of the curve order $\#E$ to the secret scalar before performing the secret scalar multiplication.

The result is preserved, $kP = (k + r\#E)P$ for a random r and no further correction is required.

Base point blinding. The second countermeasure consists of randomizing the point P into $P + R$. First the computation $k(P + R)$ is performed, and then kR is subtracted from $k(P + R)$ to yield kP. The point kR can be cheaply computed across multiple scalar multiplications.

Z-coordinate randomization. Projective coordinates represent the point (x, y) as $(X : Y : Z)$ such that $x = X/Z$ and $y = Y/Z$. Here the Z coordinate can be randomized for each execution and thus the intermediate results are randomized.

Joye and Tymen [JT01] propose two other randomizations at CHES 2001. They apply (randomly chosen) isomorphisms to perform the computation on another curve, and at the end of the computation invert the morphism.

A good overview of many attacks and countermeasures for elliptic curve cryptography can be found in the survey of Fan, Guo, de Mulder, Schaumont, Preneel and Verbauwhede [FGM+10] (see also the updated version [FV12].)

2.5.5 Particular constructions for ring-LWE

We provide the first masking of the ring-LWE algorithm in a paper published at CHES 2015 "A masked ring-LWE implementation". See Section 2.8.2 of this thesis for more details. We also published an alternate masking technique in PQCrypto 2016. See Section 2.8.3 for more details.

2.6 Higher-order masking constructions

First-order masking provides basic protection against side-channel attacks. Yet, more sophisticated, higher-order attacks have been proposed to attack first-order masking.

Definition of higher-order masking. A d-order masking is defined as a masking construction designed to withstand up to d-order attacks. A d-order attack makes use of d-order statistics. For example, a second-order DPA attack exploit information residing on the variance (and co-variance). Higher-order attacks will be studied in Section 3.1.

The construction of generic higher-order masking schemes is a very difficult problem. The constructions are very delicate. The historical track of higher-order masking

schemes is full of proposals and subsequent breaks. For example, the scheme by Schramm and Paar of 2006 [SP06] was broken one year later [CPR07], the "provably secure" scheme by Rivain and Coron of 2010 [RP10] was (academically) broken three years later [CPRR13] and the scheme of Balasch, Faust, Gierlichs and Verbauwhede published in 2012 [BFGV12] was broken in 2014. [PRR14]

2.6.1 Higher-order masked tables

Schramm–Paar table recomputation extension. Kai Schramm and Christof Paar published at CT-RSA 2006 [SP06] a masked table lookup method for Boolean masking claiming higher-order security. The showcase the authors use is AES.

Suppose the input shares are $M_0, M_1, \ldots, M_d$ and the Sbox output shares are $N_0, N_1, \ldots, N_d$ such that $N_0 \oplus N_1 \oplus \ldots \oplus N_d = S[M_0 \oplus M_1 \oplus \ldots \oplus M_d]$ Now build a table S^* such that

$$S^*[x] = S\left[x \oplus \bigoplus_{i=1}^{d} M_i\right] \oplus \bigoplus_{i=1}^{d} N_i \tag{2.12}$$

and then perform the lookup $N_0 := S^*(M_0)$. The construction of the table S^* should be performed with care. The idea is to progressively build S^* from shifted versions of S.

This countermeasure was found to be flawed by Coron, Prouff and Rivain at CHES 2007 [CPR07]. Coron et al. found a third-order flaw irrespective of the security parameter of the original scheme.

Rivain–Dottax–Prouff second-order proposals. Matthieu Rivain, Emmanuelle Dottax and Emmanuel Prouff propose in FSE 2008 [RDP08] several solutions for second-order masked Sbox lookup. They describe two methods: one is based on the table recomputation method and the other is based on comparisons.

Coron proposal. Coron proposed at EUROCRYPT 2014 [Cor14] a generic table recomputation method at any order. The construction can be related to that of Schramm and Paar. It uses different output masks and several invocations of a refreshing block. The main appealing feature of this approach is that it comes with a proof of security in the probing model (cf. Section 3.5.1).

2.6.2 ISW-derived constructions

Rivain–Prouff construction. Rivain and Prouff published at CHES 2010 [RP10] a generic method to mask the AES at any order. Their main approach is to mask

the addition and multiplication over $\mathrm{GF}(2^8)$, and express the AES computation in terms of these basic operations. For the pseudo-inversion, they use the identity $x^{-1} = x^{254}$ in $\mathrm{GF}(2^8)$ and build an addition chain using 4 multiplications.

The core of the method is a masked $\mathrm{GF}(2^8)$ multiplication. It is based on the Ishai—Sahai–Wagner construction.

This approach was shown to be (academically) flawed by Coron, Prouff, Rivain and Roche in FSE 2013 [CPRR13]. They show an attack of order $\lceil \frac{d}{2} \rceil + 1$ when the scheme is supposed to be d-th order secure. The flaw arises when composing several building blocks. Nevertheless, as the authors argue this seems to be a theoretical flaw that would be difficult to exploit in practice.

Extensions. There are many extensions of the CHES 2010 work. HeeSeok Kim, Seokhie Hong and Jongin Lim propose at CHES 2011 [KHL11] to use subfield arithmetic to compute the masked Sbox. This incurs in a substantial execution time speed-up since now many operation can be tabulated.

Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater and Matthieu Rivain generalize the Rivain—Prouff approach in an FSE 2012 publication [CGP+12] to mask any Sbox. Their approach is based on computing a k-bit Sbox lookup as the evaluation of a polynomial over $\mathrm{GF}(2^k)$. This polynomial can be derived for example by Lagrange interpolation.

2.6.3 Higher-order threshold implementations

Higher-order threshold implementations were introduced by Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov and Vincent Rijmen at ASIACRYPT 2014 [BGN+14].

Core idea. The core idea is the concept of d-th order non-completeness. A circuit is d-th order non-complete if it is possible to partition it in a way such that the union of any d sub-circuits sees at most all but one input shares.[6]

The advantage of this definition is that security in the probing model follows immediately by construction.

Constructions. Bilgin et al. give a construction for the block cipher KATAN. The only non-linear component in KATAN is a 2-bit-input AND gate. The idea is similar to first-order threshold implementations: express the function in Algebraic

[6] A related concept, although in a different context, appears in the work of Prouff and Roche at CHES 2011 [PR11].

Normal Form (sum of products), share the input bits and then rearrange the resulting terms.

Issues with composition. In the original proposal it was claimed that no additional randomness is needed during the computation, if the masked function satisfies some requirements. This would mean that the randomness requirements are much lower than for other schemes.

This was shown to induce a security flaw by Reparaz et al. at CRYPTO 2015 [RBN+15]. This is a contribution of the present thesis (see p. 151).

Improvements. We also improved on the required number of input shares to achieve a more compact implementation. For more details, see p. 47.

2.7 Theoretic considerations

2.7.1 Secret sharing

The representation mechanism of masking corresponds to secret sharing. Secret sharing was introduced in 1979 by Blakley [Bla79] and independently by Shamir in a short paper [Sha79]. A perfect secret sharing allows to split a secret in n pieces such that any $k \leq n$ pieces can be used to reconstruct the secret, and $k-1$ pieces give no information about the secret.

Shamir construction. The idea is very elegant: given k points in the plane with different abscissæ, there is only one polynomial of degree $k-1$ that passes through the k points. To share a secret value a_0, one builds a degree $k-1$ random polynomial f with $f(0) = a_0$ and distributes n points $(i, f(i))$ $(i = 1, \ldots, n)$. The polynomial f (and hence the secret a_0) can be recovered with the knowledge of any k points using interpolation.

2.7.2 Connections with multi-party computation.

Informally speaking, secure multi-party computation enables a set of parties to jointly compute a function while keeping their respective inputs private. For example, two millionaires may be interested in knowing who is wealthier without revealing their exact net worth to each other. Multiparty computation allows this kind of computation. The analogy between MPC and the side-channel problem is straightforward. Ideally, we would like to evaluate the AES function (say) when

the sensitive information is spread over several parties without revealing anything but the (final) output, in such a way that each party learns nothing more than its corresponding secret share.

The STOC 1988 Ben-Or–Goldwasser–Wigderson protocol. In a 1988 STOC publication, Ben-Or, Goldwasser and Wigderson [BOGW88] build a generic MPC protocol that computes on shares split according to Shamir's system. The abstract contains a concise description of the main result of the paper: "Every function of n inputs can be efficiently computed by a complete network of n processors in such a way that if no faults occur, no set of size $t < n/2$ of players gets any additional information (other than the function value)". The tricky part of their approach is to perform the multiplication operation. There are other claims regarding faults that are outside the scope of this thesis.

Masking based on Shamir's secret sharing. One application of the BGW protocol to design a masking scheme is given by Prouff and Roche [PR11] in CHES 2011, and independently by Goubin and Martinelli at the same conference [GM11]. Both papers instantiate the AES algorithm (see also the extended version [RP12].) For an implementation on an FPGA of an AES Sbox following the Prouff–Roche scheme see the work of Moradi and Mischke [MM13] at CHES 2013 or our work on PRESENT [CBRN14]. Masking based on Shamir's secret sharing is also called polynomial masking, and can be understood as a generalization of affine masking.

2.8 My contributions in this context

2.8.1 DPA, masking and bitslicing at 1 GHz

Problem statement. DPA is typically targeted at small- and medium-range microcontrollers typically found in e. g. smart-cards. Recently, there is a paradigm shift that moves the cryptographic functionalities traditionally running in specific hardware components into software running on powerful general purpose processors, as those found in e. g. smart-phones and tablets.

It is not clear whether DPA is still applicable to high-end processors. There are substantial differences: in high-end processors the clock speed is typically much higher (in the GHz range), they run a full-blown multitasking operating system and they feature many more peripherals (network, USB, radio).

Our contribution. In the first part of our paper published at CHES 2015 [BGRV15] we demonstrate that DPA attacks are feasible even at high clock speeds. Our

experimental platform comprises a Beaglebone Black featuring an ARM Cortex-A8 processor clocked at 1 GHz running Linux. We measure the EM emanation from a decoupling capacitor in the ground path of the main processor. The software implementation we attack is a constant-time constant-flow bitsliced AES. Although secure against timing attacks, bitslicing does not protect against DPA as expected. The main difference with lower-end platforms is that alignment and post-processing is harder.

In the second part of the paper, we tackle the issue of protecting the bitsliced implementation against DPA. Bitsliced software is based on a hardware description of the algorithm, hence, it seems reasonable to base our masking approach on hardware techniques. We use the principles of gate-level masking to implement masked versions of the XOR and AND instruction. Our masked bitsliced AES boils down to writing secure masked versions of just a handful of instructions. We reevaluate the implementation and confirm the gain in resistance against DPA attacks.

2.8.2 A masked ring-LWE implementation

Context: post-quantum cryptography. Once a quantum computer is built, Shor's algorithm [Sho97] will render most of the currently used public-key cryptography insecure. Problems based on factorization (RSA) or discrete logarithms (Diffie-Hellman, DSA and ElGamal, including elliptic curve variants) will be easy to solve on a quantum computer.

There is an entire branch of cryptography that withstand attacks even when run on a quantum computer; this branch is called *post-quantum* cryptography. Post-quantum cryptography consists of algorithms meant to be run on today's digital computers.

Context: lattice-based post-quantum cryptography. Lattices are a powerful tool for cryptographic constructions. Two advantages of lattice-based cryptography are the security guarantees often provided and the efficiency of the constructions. Some proposals offer strong security reductions to computationally hard problems, and are relatively efficient to execute. However, key sizes are usually larger than their established, non-quantum secure alternatives, yet still manageable for many applications.

Problem statement. Post-quantum cryptography is still at its infancy. There are already some proposals, but they are not as mature as deployed public-key systems (RSA or ECC). Post-quantum schemes do not inherently protect against side-channel analysis. There is thus the need to design efficient and side-channel secure post-quantum public-key crypto-systems.

Our research is focused on designing a masked ring-LWE public-key decryption function. Ring-LWE is a promising and versatile cryptographic primitive. We focus on the decryption operation since it handles long-term secrets susceptible to DPA.

Our contribution. In our paper at CHES 2015 [RRVV15] we present a masked ring-LWE public-key decryption function. Our solution consists of two main components: the first one is essentially arithmetic masking to perform polynomial arithmetic operations; the second one is a custom masked decoder that performs a threshold operation.

Arithmetic masking of the polynomial operations allows to recycle the arithmetic co-processor of an unprotected implementation, making it easy to upgrade an unmasked implementation to a masked one. (In our own design, we base the arithmetic co-processor on Sujoy Sinha Roy's CHES 2014 design [RVM+14].)

The bespoke decoder performs a thresholding operation in the masked domain. The approach is original and not based on a previous design. The masked decoder is probabilistic and the operation has to be repeated 16 times to achieve a negligible impact on the system error rate. It outputs Boolean shares suitable for derivation of, e. g. an AES-256 key (hybrid encryption).

Our design is not only theoretical. We implemented the whole design on a Virtex-II FPGA. The overhead in area is quite small; in cycles the masked implementation requires about $\times 2.7$ more time. We also verified the practical gain in security against DPA attacks.

Follow-up work. An extended version of this paper was published in a special issue of the Journal of Cryptographic Engineering (by invitation) [RRdC+16]. The extension consists of a software implementation for ARM. This was performed with the additional collaboration of Ruan de Clercq.

Another follow-up work is our publication in PQCrypto 2016 [RdCR+16] that we describe next.

2.8.3 Additively homomorphic ring-LWE masking

Our contribution. In our PQCrypto 2016 paper [RdCR+16], we present a new masking scheme for ring-LWE decryption. Our scheme exploits the additively-homomorphic property of the existing ring-LWE encryption schemes and computes an additive-mask as an encryption of a random message. Our solution differs in several aspects from the CHES 2015 approach; most notably we do not require a masked decoder but work with a conventional, unmasked decoder. Hence, we can secure a ring-LWE implementation using additive masking with minimal changes.

Our masking scheme is also very generic in the sense that it can be applied to other additively-homomorphic encryption schemes, and can be viewed as ciphertext blinding.

2.9 Conclusion

We have seen that the design space of a masking schemes is very large. There are many different approaches that can be followed to mask a given cryptographic algorithm. We have presented an overview of these approaches.

Our contribution to the design of masking schemes are three new masking approaches: a generic software approach tailored to a new platform (high-performance embedded processors) based on gate-level masking and two proposals for a recently-proposed public-key algorithm (ring-LWE).

History shows that the design of masking schemes is a delicate task. In the next chapter, we tackle the issue of analyzing masking schemes.

Chapter 3

Analysis of masking schemes

In this chapter we deal with attacks on masking in Section 3.1 and with tools for evaluating masked implementations in Section 3.2.

3.1 Attacks on masking

In this section we describe how a masked implementation can still be attacked. However, it should not come as a surprise that attacking masked implementations is hard, as that is precisely the whole point of masking. Rather than aiming at unconditional security, a proper masked implementation makes attacks highly impractical.

3.1.1 First-order attacks

Definition of first-order attacks. By first-order attacks we mean DPA attacks that exploit information residing in on the mean power consumption. There are many flavors of first-order DPA attacks: from Kocher's single-bit DPA, to Brier–Clavier–Olivier's Correlation Power Analysis and many other variants.

In theory, first-order masking perfectly protects against first-order attacks. In practice, a properly implemented and sound masking scheme makes first-order attacks impractical. Hence, first-order attacks are often carried out to verify that the masking was implemented properly.

3.1.2 Second-order attacks

Second-order attacks are the natural extension of first-order attacks to break masking. We define second-order attacks as those that exploit information in the variance or covariance (second-order statistical moments). Second-order attacks were already suggested in the original DPA paper.

Simple example. We describe here a second-order attack against a first-order masked implementation. Suppose the first-order masked implementation handles the two shares V_1 and V_2 of a certain intermediate V such that $V = V_1 \oplus V_2$ at time sample t_1 and t_2 respectively. A second-order attack combines the measurement data L at timesamples t_1 and t_2 through a certain combination function c, for instance $c(L(t_1), L(t_2)) = L(t_1) \cdot L(t_2)$, and then proceeds to correlate $c(L(t_1), L(t_2))$ against key-dependent predictions for $L(V)$. Note that while each $L(t_1)$ or $L(t_2)$ is individually independent from V, the product $L(t_1) \cdot L(t_2)$ is not.

Properties of second-order attacks. While the procedure for second-order attacks resembles first-order attacks (bar the pre-processing step), the number of traces required to perform a second-order attack is substantially larger than the number of traces that one would require to perform a first-order attack against an unprotected implementation at the same noise level. This is the whole point of masking. Asymptotically, the number of required traces grows as per σ^d, where σ is the noise level and d is the order of the attack.

Additionally, the adversary normally does not know in advance the exact location of the timesample pair (t_1, t_2) corresponding to the handling of each share. Thus, if the implementation handles the shares in different time instants, the second-order attack has to be repeated over (potentially) all combinations of time sample pairs. This incurs a significant computation burden.

Combination functions. There are many proposals for combination functions. Two proposals are commonly used in practice: centered product and absolute difference. Prouff and Rivain proved, under a certain model, that the centered product with the correlation coefficient distinguisher is optimal (yields maximum correlation) [PRB09]. Another pre-processing function is the absolute difference, as proposed by Messerges [Mes00b].

Higher-order attacks. In the same spirit as second-order attacks, it is possible to generalize and conceive third- or higher-order attacks. A d-th order attack exploits information located in the d-th order statistical moment. Thus, higher-order attacks are very sensitive to noise and become quickly impractical as the order increases.

Higher-order attacks can be used to break higher-order masking schemes. A d-th order masking is designed to withstand up to d-th order attacks. In theory, a d-th order masking can always be broken with a $(d+1)$-th order attack.

3.1.3 Biasing the masks

Tiri and Schaumont [TS06] describe in 2006 the folding attack. This attack is a two-stage procedure: first the attacker derives the mask value from each measurement and classifies the measurements according to their mask value. Secondly, a first-order DPA attack is performed on "folded" measurements. (Alternatively, a first-order DPA attack can be performed on a subset of measurements.) A similar idea is presented by Jaffe in 2006 [Jaf06].

This attack requires that the mask has a noticeable impact on each individual measurement. Thus, measurements should contain enough noise so that individual shares are not directly visible on each measurement.

The work of Pan, den Hartog and Lu [PdHL09] focuses on masked software implementations and follows the previous two-step procedure. However, the first step is more elaborate and requires an additional horizontal DPA attack to recover the mask.

3.2 Evaluating masking

One possible strategy to evaluate whether a masked implementation achieves the security goals is to perform key-recovery attacks on each intermediate variable. This can be an expensive process. An alternate strategy is to use leakage detection tests.

3.2.1 Leakage detection tests

Leakage detection tests can be traced back to a 2000 publication of Jean-Sébastien Coron, David Naccache and Paul C. Kocher. [CKN00] (See also a journal version by the same authors [CNK04].) Recently leakage detection techniques have been pushed for use in evaluation contexts by CRI [GJJR11], see also the manuscript [CDG+13].

Leakage detection tests have a different goal than key-extracting DPA or SPA attacks. Leakage detection does not focus on extracting keys, but on assessing whether there is leakage or not. The presence of leakage is a necessary, yet not sufficient, condition for key-extracting attacks to work. Roughly speaking, if an implementation fails a leakage detection test, there is a dependency between the

side-channel signal and the data being processed. This dependency is likely to be useful for key extraction. On the other hand, when an implementation passes a leakage detection test, no data dependency is detected, and hence key-recovery attacks will not work.

The outcome of a leakage detection test depends on the number of measurements available. This means in particular that if more measurements are available, leakage can start to be detectable. In other words, leakage detection tests detect leakage "on the strength of the evidence presented" [CNK04].

Typical construction of a leakage detection test. In a typical leakage detection test, the evaluator takes two sets of N measurements each. Measurements from one set correspond to the (unmasked) plaintext value p_1 and the other set to the plaintext value $p_2 \neq p_1$. The collected measurements are input to statistical hypothesis testing methods to check for differences in the two conditional distributions.

A common null hypothesis H_0 is

H_0: the mean power consumption of the two sets is the same.

If this hypothesis is true, the implementation is not vulnerable to first-order DPA attacks. This hypothesis can be tested with the t-test [Stu08, Wel47]. This tests for first-order security.

In the next paragraphs, an intermediate variable is *active* if the value it takes in the two sets is different.

Specific and non-specific tests. Non-specific tests do not assume much on the architecture or leaking behavior of the device. The plaintexts p_i are chosen so that all intermediate variables in all rounds are active. That is, the whole state is active.

Specific tests choose plaintexts to target and activate specific intermediate variables. The advantage is that the evaluator gains more insight on what is leaking. Normally the statistics converge faster and thus less traces are required. Of course, this requires that one makes assumptions on the architecture of the implementation. Furthermore, specific tests require to consciously craft the plaintext values p_i. For an exemplary list of input vectors for RSA we refer to the work of Jaffe, Rohatgi and Witteman [JRW11].

Fix vs. random or fix vs. fix. One way to make fewer assumptions on the leakage behavior of a device is to randomize the plaintext value of one set. This results in a fix vs. random test. If the practitioner is willing to make certain assumptions on the leakage behavior, he may speed up the evaluation process by performing a fix vs. fix test, as noted by Durvaux and Standaert [DS16].

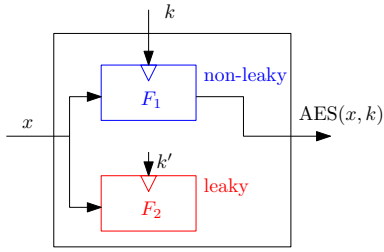


Figure 3.1 – Thought experiment for leakage detection.

Semifix vs. random. An inconvenience of non-specific tests is that they might detect leakage from non-exploitable intermediate variables such as input plaintext or output ciphertext. One can get rid of those if it is possible to set and change the key on the device. (In this situation, one would also detect leakage from the key schedule.) If this functionality is not available, one can resort to semifix vs. random tests.

A semifix vs. random test randomizes the values from one set of measurements, and for the other set it chooses plaintexts values from a "semifix" set. This set of plaintexts is defined so that only a significant portion of the state is active in a chosen inner round.

Higher-order extensions. Leakage detection tests can naturally be applied to test leakage at any statistical order, by adequately pre-processing the measurements. In this way, higher-order security claims of masking schemes can be verified.

Leakage detection vs. key extraction. Note that a leakage-detection test says nothing about the feasibility of extracting keys. This is very easy to see in the following thought experiment. Suppose the following construction in Figure 3.1: on input x a box outputs the value $\mathrm{AES}(x, k)$ where k is a secret key. This value is computed with a perfectly non-leaky hardware implementation F_1. In parallel, the value $\mathrm{AES}(x, k')$ is computed (but the output discarded) with leaky hardware F_2. The second key k' is different and independent of k, and changes whenever k is updated.

Here it is easy to see that a leakage detection test on the construction of Figure 3.1 will definitely fail (that is, detect leakage): by hypothesis F_2 is leaky and this will be detected. However, the computation of F_2 is independent of the secret k, and the only computation depending on k is non-leaky. Thus, no key-recovery attack can succeed.

3.3 Complications of masking

Masking is notoriously hard to implement in practice, both in hardware and in software. In hardware, glitches can lead to exploitable first-order leakage, while in software, the distance leakage behavior of registers and other components can as well lead to defective masked implementations.

3.3.1 Glitches

History. Mangard, Popp and Gammel raised the issue at CT-RSA 2005 [MPG05] that glitches occurring in standard CMOS may render masking insecure. They theoretically derived their results and backed up their claims with simulations. A bit later, Mangard, Pramstaller and Oswald at CHES 2005 [MPO05] presented results attacking an actual masked hardware AES with the same principles.

What are glitches? A glitch is, in static CMOS, a spurious transition of nodes in a combinational circuit within one clock cycle, resulting from different arrival times of the input signals.

Solutions. The mitigation of glitches is a well-studied problem in digital design, since they are not only inconvenient from a security point of view. Glitches are useless transitions that consume extra energy and thus digital designers tend to minimize them to achieve low-power and high-speed circuits. There are strategies to reduce glitches (e.g., balancing the path delay using combinational tree-like structures) or fully eliminate them (e.g. using dynamic logic styles, such as Domino or dynamic differential such as SABL [TAV02b] or WDDL [TV04b]).

There have been proposed many other solutions to cope with glitches in hardware. See Section 2.4.2 for a glitch-conscious set of masked gates due to Infineon and Section 2.4.3 for threshold implementations, a masking technique that aims at providing security in the presence of glitches.

3.3.2 Distance issues

What is the problem? A careless software masked implementation may schedule to store one share V_1 in a certain register R_1 and immediately later store the second share V_2 in the same register R_1. This is bad for security.

The power consumption of a register is strongly related to the Hamming distance between consecutive values held in the register. Hence, the power consumption

may be strongly related to the unmasked value $V = V_1 \oplus V_2$, rendering masking useless.

This kind of register allocation may be produced by a compiler. This is a reason why masking is normally implemented in the lowest-level language to achieve maximum control.

The same effect can occur within other hardware components of the processor (for example, buses), and highly depends on the architecture and implementation of the processor. These details, critical for writing a proper masked implementation, are usually not publicly documented in commercial off-the-shelf microcontrollers.

Solutions. The canonical solutions are to load a third, unrelated value in between the two shares, for example a constant or a random value.

We also explore another solution in our publication "On the cost of lazy engineering (for software masked implementations)" [BGG+14].

3.4 Randomness

Randomness is an essential part of masking. Indeed, if it is possible to tamper with the randomness source, a masked implementation will not deliver its expected security guarantees. In the extreme case, switching off completely the randomness source will switch off completely the masking countermeasure.

Properties. There are some differences between the properties expected from a RNG suitable for masking and a RNG for producing cryptographic keys. For masking, the randomness source is expected to have good statistical properties. However, for masking the randomness source needs not to be a cryptographically strong RNG. This is due to the fact that no part of the random stream of masks leaves the device. That is, all masks are secret to the adversary. Thus, one typically needs a RNG that

- fulfills statistical requirements, and
- if using a PRNG, its initial entropy is large enough to stop attacks based on guessing the state.

How much randomness? It is hard to give bounds on how much randomness is actually needed for a secure masked execution. A single-bit mask for the whole circuit is theoretically sufficient to withstand first-order attacks, if implemented properly.

Note that the RNG itself is a security-critical design, and should be protected against manipulation from many other attacks. See for example the work of Markettos and Moore at CHES 2009 [MM09].

3.5 Theoretic considerations

3.5.1 Probing model

Underlying concept. The probing model results from an abstraction of probing attacks. Probing attacks are invasive attacks that place microscopic probes on top of the integrated circuit to eavesdrop internal signals from the chip. It is not hard to see that probing attacks have devastating consequences for security: it suffices to eavesdrop the bus that transports the key to break the whole system. Having access to a single bit of intermediate variables is usually also enough to break either secret-key or public-key algorithms, as shown by Handschuh, Paillier and Stern in 1999 [HPS99]. (Modern security integrated circuits feature countermeasures that make probing attacks harder, such as active shields, bus scrambling and glue logic.)

Masking in the probing model. Roughly speaking, a masking scheme secure in the t-probing model means that even if the adversary has access to (any) t intermediate variables appearing during the computation, the adversary learns nothing.

A rough sketch of proofs in the probing model goes as follows. One proves that the distribution of any t probe measurements can be simulated even without knowing the secrets. Hence, probing provides no information about the secret. These proofs normally build a simulator that emulates the behavior of the masked circuit, with the exception that the simulator does not carry secrets, yet is able to emulate the actual circuit.[1]

One advantage of the probing model is that the proofs are concrete. In contrast to the noisy leakage model (cf. Section 3.5.2), the proofs hold for any noise level and do not depend on the asymptotic behavior of certain parameters. An instructive application of the probing model appears in the private circuits paper [ISW03]. The proof is very elegant.

A word of warning on proofs in the probing model. The probing model, however, is deceptively simple. There are "proofs" in the probing model that turned out to

[1] Simulation based proofs have their foundations on the identity of indiscernibles, attributed to Leibniz.

be flawed, either because of technical reasons or incorrect modeling. See Section 2.6 for references of flawed proofs.

3.5.2 Noisy leakage model

The original proof of masking is given in the noisy leakage model. This model assumes that the adversary is able to observe a noisy version of the leakage behavior function of the device. What was proven in [CJRR99] is that it is exponentially difficult to distinguish the distribution of a masked bit as the masking order increases. More precisely, the amount of samples required to distinguish the two distributions (corresponding to the two possible values for the unshared bit) grows as σ^d, where σ denotes the noise level. The result, however, is asymptotic as $\sigma \to \infty$.

3.5.3 Connections between them

In a way, the d-probing model loosely resembles a d-variate DPA attack. The analogy is not exact, since there are ways to adapt a d-variate leakage to a univariate leakage, for example by pre-processing the measurement traces.[2]

The probing and noisy leakage models turn out to be theoretically equivalent. For more details, see the work of Sebastian Faust.

3.5.4 On provable security

Provable security aims at providing security proofs for cryptographic constructions. The ultimate goal is to provide constructions that cannot be broken under a given model. The concept of provable security is, without any doubts, laudable. It is applied not only to secure implementations of cryptographic algorithms but to all aspects of cryptography (for example, protocols).

The practice of provable security is a controversial topic in cryptography in general. Provable security often gives a sense of psychological relief, but in many cases this is misleading. The apparent mathematical rigour may hinder the healthy scientific questioning process that allows to break schemes in new ways.

There are several reasons that limit the practical significance of security proofs in our view. Often, security proofs are just flawed. This is understandable since security proofs can be very complex constructions that are hard to grasp or verify

[2]This pre-processing may be performed by the own measurement setup (compression/convolution).

even for experts in the field. Another reason is that the model taken in consideration for writing the proof may not fully correspond to the actual implementation.

It is our view that security proofs may be a useful tool to spend more time analyzing a side-channel countermeasure, but should not be considered as the unique argument to judge the effectiveness of a countermeasure.

3.6 My contributions in this context

3.6.1 Selecting time samples for multivariate DPA attacks

Problem statement. When attacking masked implementations, a common assumption is that the evaluator knows the exact time samples corresponding to the handling of several shares, so that he can feed those time samples into the chosen higher-order DPA attack technique. This information is important since otherwise the attack has to be iterated over all combinations of time samples, leading to a combinatorial explosion in the computational requirements.

Prior work. This information is easy to get if the evaluator has access to the code, or to an open sample that he can profile. In the general case, however, the problem is much harder. Prior methods are highly heuristic or work under tight assumptions.

Our contribution. Our publication at CHES 2012 [RGV12] solves this problem in a systematic fashion (it will "always" output all exploitable tuples of time samples that exist, given enough measurements), black-box conditions (no access to the code nor implementation details are needed), under any leakage model (although the evaluator can benefit if he knows details about the leakage behavior).

Our solution leads to a constant, yet substantial, speed-up in the computational effort required to carry out multivariate higher-order DPA attacks. In our experiments, we measured improvements of two orders of magnitude, bringing attacks that needed two weeks of computation to less than one day.

The solution is based on computing the mutual information between tuples of traces and plaintext. There is no key guess involved, yet our method will output tuples corresponding to exploitable time sample such as Sbox inputs and outputs. In retrospect, our method can be understood as a multivariate leakage detection test at any order.

Follow-up work. We have extended the idea to exploit the information laying on concrete statistical moments. This reduces the generality of the approach, but vastly increases its performance. The work is under revision pending submission.

3.6.2 Consolidating masking schemes

Our contribution. In our paper at CRYPTO 2015 [RBN+15] we point out several similarities between highly theoretical and practical masking schemes. These observations allow to compare side-by-side different masking schemes. In this comparison, subtle differences arise.

We extend the ISW algorithm to compute other more complicated functions, and to bring protection even in the presence of glitches.

These differences allow us to break higher-order threshold implementations as originally presented. We point out an issue with the composition of shared functions and demonstrate in a proof-of-concept that higher-order threshold implementations are not actually higher-order secure. We also give directions for mitigating this inconvenience.

On the bright side, our comparative analysis also allows us to bring improvements from one masking scheme to the other. We present threshold implementations that use significantly less input shares (from $td + 1$ to $d + 1$, where t is the algebraic degree of the function and d the security level.)

We point out also an application of the idea of non-completeness to ISW-based software schemes. This can provide protection against distance-based leakage behaviors.

Follow-up work. This work has been extended to give a higher-order masked threshold implementation of the AES Sbox using $d + 1$ shares with the additional collaboration of Thomas de Cnudde. We present first- and second-order secure implementations of the AES Sbox. Practical experiments with 100 million traces confirm the validity of our approach. This results in the smallest up to date threshold implementation of the AES Sbox. This work is under submission.

3.6.3 Detecting flawed masking schemes with leakage detection tests

Problem statement. Many proposed masking schemes, even carrying "security proofs", are eventually broken because they are flawed by design. The security validation process is nowadays a lengthy, tedious and manual process.

Our contribution. In our FSE 2016 publication [Rep16] we report on a method to verify the soundness of a masking scheme before implementing it on a device. We show that by instrumenting a high-level implementation of the masking scheme and by applying leakage detection techniques, a system designer can quickly assess at design time whether the masking scheme is flawed or not, and to what extent. (Note that one can still implement a sound masking scheme in an insecure fashion, and thus this tool is no magic bullet.)

Our method requires not more than working high-level source code and is based on simulation. Thus, our method can be used already in the very early design stages. We validate our approach by spotting in an automated fashion first-, second- and third-order flaws in recently published state-of-the-art schemes in a matter of seconds with limited computational resources. We also present a new second-order flaw in a table recomputation scheme, and show that the approach is useful when designing a hardware masked implementation.

Chapter 4

Conclusion and future work

Predictions are hard (especially about the future). Nevertheless, we speculate in the following on the future research developments of masking.

Masking in retrospective. Masking is a mature area of research that has received over 15 years of continuous research effort. This is exacerbated by intense industrial interest. Thus, one could think that the topic lacks new ideas. We think that, on the contrary, masking will continue being a fertile area of research.

On the one hand, it is clear that masking is a very delicate topic, where even experts propose schemes that later turn out to be flawed. Thus, a first area of future research is a thorough study of newly proposed masking schemes. There are probably bugs in schemes that are already published, and a detailed study of their relevance in practice would be desirable.

New masking schemes. On the other hand, there is room to develop more efficient schemes for current algorithms. The primary reason may be economic: there is a clear incentive from industry to optimize masking schemes in order to reduce performance overheads.

On the more academic side, some masking schemes have been proposed focusing not exclusively on performance but aiming at stronger security guarantees (beyond Boolean masking). We think this will be another avenue of research.

Randomness. A study of the required randomness for masking schemes in terms of quality and quantity would be very useful. Higher-order schemes usually are very expensive in terms of randomness, but concrete bounds on the amount of

randomness are fully understood yet. We leave as future work masking schemes optimized for low entropy usage.

Masking on new platforms. It is inevitable that future cryptographic algorithms run on different platforms than we know today. A current trend is to move the cryptographic operations from dedicated hardware to powerful generic microprocessors (such as phones and personal electronic gadgets.) This is done in Host Card Emulation. However, the security requirements will probably stay without major changes: side-channel protection will still be an issue.

The attack principles for side-channel security may well stay without major modifications. It was refreshing to see that the DPA attack against the ARM Cortex A8 used the same principles as DPA attacks on older platforms; and an interesting fact was that the established principles of gate-level masking could be recycled to mask the software implementation running on a new platform.

Masking in combination with other countermeasures. It will become very interesting to design countermeasures that are effective against a wide range of attacks. When the platform of choice progressively moves from hardware to software, tamper resistance may require to implement software protection mechanisms such as obfuscation and white-box cryptography. The interplay between these is not yet fully understood.

Masking on new algorithms. We believe that a migration towards post-quantum algorithms in established protocols (such as EMV and TLS) will happen in the near future. Those protocols will require secure implementation of post-quantum algorithms that are also resistant against (among others) side-channel attacks. Our first contribution in this field is the masking of ring-LWE; we believe that further improvements will yield even more compact implementations.

Bibliography

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[AG03] Mehdi-Laurent Akkar and Louis Goubin. A Generic Protection against High-Order Differential Power Analysis. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 192–205. Springer, 2003.

[And08] Ross J Anderson. *Security engineering - a guide to building dependable distributed systems (2. ed.)*. Wiley, 2008.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BDPV10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Building power analysis resistant implementations of Keccak. Second SHA-3 candidate conference, August 2010.

[Ber04] Daniel J. Bernstein. Cache-timing attacks on AES, 2004. URL: `http://cr.yp.to/papers.html#cachetiming`.

[BFGV12] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and Practice of a Leakage Resilient Masking Scheme. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on*

the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings, volume 7658 of *Lecture Notes in Computer Science*, pages 758–775. Springer, 2012.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BGRV15] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1 GHz. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2015.

[Bil15] Begül Bilgin. *Threshold Implementations As Countermeasure Against Higher-Order Differential Power Analysis*. PhD thesis, KU Leuven and University of Twente, Centre for Telematics and Information Technology, Enschede, The Netherlands, may 2015.

[Bla79] G R Blakley. Safeguarding cryptographic keys. In Richard E Merwin, Jacqueline T Zanca, and Merlin. Smith, editors, *1979 National Computer Conference: June 4–7, 1979, New York, New York*, volume 48 of *AFIPS Conference proceedings*, pages 313–317, 1979.

[BNN+12] B Bilgin, S Nikova, V Nikov, V Rijmen, and G Stütz. Threshold Implementations of all 3×3 and 4×4 S-boxes. Cryptology ePrint Archive, Report 2012/300, 2012.

[Boa73] David G Boak. *A History of U.S. Communications Security; the David G. Boak Lectures*, volume I. National Security Agency, 1973.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Compu-

tation (Extended Abstract). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[CBRN14] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, and Svetla Nikova. Higher-Order Glitch Resistant Implementation of the PRESENT S-Box. In Berna Ors and Bart Preneel, editors, *Cryptography and Information Security in the Balkans - First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers*, volume 9024 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2014.

[CDG+13] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) methodology in practice. International Cryptographic Module Conference, 2013.

[CGP+12] Claude Carlet, Louis Goubin, Emmanuel Prouff, Michaël Quisquater, and Matthieu Rivain. Higher-Order Masking Schemes for S-Boxes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2012.

[Cha82] David Chaum. Blind Signatures for Untraceable Payments. In David Chaum, Ronald L Rivest, and Alan T Sherman, editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 199–203. Plenum Press, New York, 1982.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CKN00] Jean-Sébastien Coron, Paul C Kocher, and David Naccache. Statistics and Secret Leakage. In Yair Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2000.

[CNK04] Jean-Sébastien Coron, David Naccache, and Paul C Kocher. Statistics and secret leakage. *ACM Trans. Embedded Comput. Syst.*, 3(3):492–508, 2004.

[Con08] EMVCo Consortium. Integrated Circuit Card. Specifications for Payment Systems. Book 2. Security and Key Management. Technical report, jun 2008.

[Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[Cor14] Jean-Sébastien Coron. Higher Order Masking of Look-Up Tables. In Phong Q Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-Order Side Channel Security and Mask Refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template Attacks. In Burton S Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 13–28. Springer, 2002.

[CRV14] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast Evaluation of Polynomials over Binary Finite Fields and Application to Side-Channel Countermeasures. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 170–187. Springer, 2014.

[CRV15] Jean-Sébastien Coron, Arnab Roy, and Srinivas Vivek. Fast evaluation of polynomials over binary finite fields and application to side-channel countermeasures. *J. Cryptographic Engineering*, 5(2):73–83, 2015.

[DK10] Ivan Damgård and Marcel Keller. Secure Multiparty AES. In Radu Sion, editor, *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, January 25-28, 2010, Revised Selected Papers*, volume 6052 of *Lecture Notes in Computer Science*, pages 367–374. Springer, 2010.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[FG05] Wieland Fischer and Berndt M Gammel. Masking at Gate Level in the Presence of Glitches. In Josyula R Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 187–200. Springer, 2005.

[FGM+10] Junfeng Fan, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel, and Ingrid Verbauwhede. State-of-the-art of Secure ECC Implementations: A Survey on Known Side-channel Attacks and Countermeasures. In Jim Plusquellic and Ken Mai, editors, *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*, pages 76–87. IEEE Computer Society, 2010.

[FMD07] Guillaume Fumaroli, Emmanuel Mayer, and Renaud Dubois. First-Order Differential Power Analysis on the Duplication Method. In K Srinathan, C Pandu Rangan, and Moti Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 210–223. Springer, 2007.

[Fri72] Jeffrey Friedman. TEMPEST: A signal problem. *NSA Cryptologic Spectrum*, 1972.

[FV12] Junfeng Fan and Ingrid Verbauwhede. An Updated Survey on Secure ECC Implementations: Attacks, Countermeasures and Cost. In David Naccache, editor, *Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, volume 6805 of *Lecture Notes in Computer Science*, pages 265–282. Springer, 2012.

[FWA+13] Dirk Feldhusen, Guntram Wicke, Arnold Abromeit, Lex Schoonen, and BSI. Minimum Requirements for Evaluating Side-Channel Attack Resistance of RSA, DSA and Diffie-Hellman Key Exchange Implementations. Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2013. Part of AIS 46.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. NIST non-invasive attack testing workshop, 2011.

[GM04] Jovan Dj. Golić and R Menicocci. Universal Masking on Logic Gate Level. *Electronics Letters*, 40(9):526–528, apr 2004.

[GM11] Louis Goubin and Ange Martinelli. Protecting AES with Shamir's Secret Sharing Scheme. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2011.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[GT02] Jovan Dj. Golić and Christophe Tymen. Multiplicative Masking and Power Analysis of AES. In Burton S Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 198–212. Springer, 2002.

[HPS99] Helena Handschuh, Pascal Paillier, and Jacques Stern. Probing Attacks on Tamper-Resistant Devices. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 303–315. Springer, 1999.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[Jaf06] Josh Jaffe. More Differential Power Analysis: Selected DPA Attacks. ECRYPT Summerschool on Cryptographic Hardware, Side Channel and Fault Analysis, 2006.

[JRW11] Josh Jaffe, Pankaj Rohatgi, and Marc Witteman. Efficient side-channel testing for public key algorithms: RSA case study. NIST non-invasive attack testing workshop, 2011.

[JT01] Marc Joye and Christophe Tymen. Protections against Differential Analysis for Elliptic Curve Cryptography. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 377–390. Springer, 2001.

[Kah96] David Kahn. *The codebreakers: the story of secret writing.* Scribner, New York, 1996.

[KHL11] HeeSeok Kim, Seokhie Hong, and Jongin Lim. A Fast and Provably Secure Higher-Order Masking of AES S-Box. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 95–107. Springer, 2011.

[KJJ98a] Paul C Kocher, Joshua M Jaffe, and Benjamin C Jun. Cryptographic computation using masking to prevent differential power analysis and other attacks, 1998. US Patent 7,668,310.

[KJJ98b] Paul C Kocher, Joshua M Jaffe, and Benjamin C Jun. DES and other cryptographic, processes with leak minimization for smartcards and other cryptosystems, 1998. US Patent 6,278,783.

[KJJ98c] Paul C Kocher, Joshua M Jaffe, and Benjamin C Jun. Introduction to Differential Power Analysis and Related Attacks. Technical report, Cryptography Research, 1998.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume

1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KKG02] Franz Klug, Oliver Kniffler, and Berndt Gammel. Rechenwerk, Verfahren zum Ausführen einer Operation mit einem verschlüsselten Operanden, Carry-Select-Addierer und Kryptographieprozessor, 2002. DE Patent 10,201,449.

[KLL+11] Wolfgang Killmann, Tanja Lange, Manfred Lochter, Wolfgang Thumser, and Guntram Wicke. Minimum Requirements for Evaluating Side-Channel Attack Resistance of Elliptic Curve Implementations. Technical report, Bundesamt für Sicherheit in der Informationstechnik, 2011. Part of AIS 46.

[Koc96] Paul C Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[Kuh02] Markus G Kuhn. Optical Time-Domain Eavesdropping Risks of CRT Displays. In *2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 12-15, 2002*, pages 3–18. IEEE Computer Society, 2002.

[Kuh03] Markus G Kuhn. Compromising emanations: eavesdropping risks of computer displays. Technical Report UCAM-CL-TR-577, University of Cambridge, Computer Laboratory, dec 2003.

[LU02] Joe Loughry and David A Umphress. Information leakage from optical emanations. *ACM Trans. Inf. Syst. Secur.*, 5(3):262–289, 2002.

[Mes00a] Thomas S Messerges. Securing the AES Finalists Against Power Analysis Attacks. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2000.

[Mes00b] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç K Koç and C Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *LNCS*, pages 238–251. Springer, 2000.

[MM09] A. Theodore Markettos and Simon W. Moore. The frequency injection attack on ring-oscillator-based true random number generators. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware*

and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings, volume 5747 of *Lecture Notes in Computer Science*, pages 317–331. Springer, 2009.

[MM13] Amir Moradi and Oliver Mischke. On the Simplicity of Converting Leakages from Multivariate to Univariate - (Case Study of a Glitch-Resistant Masking Scheme). In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2013.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[MPL+11] A Moradi, A Poschmann, S Ling, C Paar, and H Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully Attacking Masked AES Hardware Implementations. In Josyula R Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[MvOV96] Alfred Menezes, Paul C van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[NRS09] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Non-linear Functions in the Presence of Glitches. In *Information Security and Cryptology – ICISC 2008*, volume 5461 of *LNCS*, pages 218–234. Springer, 2009.

[NRS11] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. *J. Cryptology*, 24(2):292–321, 2011.

[Nyb93] Kaisa Nyberg. Differentially Uniform Mappings for Cryptography. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 1993.

[OMP04] Elisabeth Oswald, Stefan Mangard, and Norbert Pramstaller. Secure and Efficient Masking of AES - A Mission Impossible? *IACR Cryptology ePrint Archive*, 2004:134, 2004.

[OMPR05] Elisabeth Oswald, Stefan Mangard, Norbert Pramstaller, and Vincent Rijmen. A Side-Channel Analysis Resistant Description of the AES S-Box. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, volume 3557 of *Lecture Notes in Computer Science*, pages 413–423. Springer, 2005.

[PdHL09] Jing Pan, J I den Hartog, and Jiqiang Lu. You Cannot Hide behind the Mask: Power Analysis on a Provably Secure \emphS-Box Implementation. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications, 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, volume 5932 of *Lecture Notes in Computer Science*, pages 178–192. Springer, 2009.

[PMK+11] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-Channel Resistant Crypto for Less than 2,300 GE. *Journal of Cryptology*, 24(2):322–345, 2011.

[PR11] Emmanuel Prouff and Thomas Roche. Higher-Order Glitches Free Implementation of the AES Using Secure Multi-party Computation Protocols. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[PRR14] Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. On the Practical Security of a Leakage Resilient Masking Scheme. In

Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2014.

[Rab96] Jan M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[RdCR+16] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-lwe masking. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2016.

[RDP08] Matthieu Rivain, Emmanuelle Dottax, and Emmanuel Prouff. Block Ciphers Implementations Provably Secure Against Second Order Side Channel Analysis. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2008.

[Rep16] Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In Thomas Peyrin, editor, *Fast Software Encryption, 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016*, volume 0000 of *Lecture Notes in Computer Science*, page 20. Springer, 2016.

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20,*

2010. Proceedings, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[RP12] Thomas Roche and Emmanuel Prouff. Higher-order glitch free implementation of the AES using Secure Multi-Party Computation protocols - Extended version. *J. Cryptographic Engineering*, 2(2):111–127, 2012.

[RRdC+16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015.

[RV13] Arnab Roy and Srinivas Vivek. Analysis and Improvement of the Generic Higher-Order Masking Scheme of FSE 2012. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 417–434. Springer, 2013.

[RVM+14] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. In *CHES*, volume 8731 of *LNCS*, pages 371–391. Springer, 2014.

[Sha49] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sha99] A Shamir. Protecting smart cards from power analysis with detachable power supplies, 1999. US Patent 6,507,913.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.*, 26(5):1484–1509, October 1997.

[SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose,*

CA, USA, February 13-17, 2006, Proceedings, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[SS06] Daisuke Suzuki and Minoru Saeki. Security evaluation of DPA countermeasures using dual-rail pre-charge logic style. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2006.

[Stu08] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[SWP03] Kai Schramm, Thomas J Wollinger, and Christof Paar. A New Class of Collision Attacks and Its Application to DES. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 206–222. Springer, 2003.

[TAV02a] K Tiri, M Akmal, and I Verbauwhede. A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *Solid-State Circuits Conference, 2002. ESSCIRC 2002. Proceedings of the 28th European*, pages 403–406, sep 2002.

[TAV02b] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dyamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *ESSCIRC*, pages 403–406, 2002.

[Tho01] Larry Puhl Thomas S. Messerges Ezzat A. Dabbish. Method and apparatus for preventing information leakage attacks on a microelectronic assembly, 2001. US Patent 6,295,606.

[TKL04] Elena Trichina, Tymur Korkishko, and Kyung-Hee Lee. Small Size, Low Power, Side Channel-Immune AES Coprocessor: Design and Synthesis Results. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *Advanced Encryption Standard - AES, 4th International Conference, AES 2004, Bonn, Germany, May 10-12, 2004, Revised Selected and Invited Papers*, volume 3373 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2004.

[Tri03] Elena Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.

[TS06] Kris Tiri and Patrick Schaumont. Changing the Odds Against Masked Logic. In Eli Biham and Amr M Youssef, editors, *Selected Areas*

in Cryptography, 13th International Workshop, SAC 2006, Montreal, Canada, August 17-18, 2006 Revised Selected Papers, volume 4356 of *Lecture Notes in Computer Science*, pages 134–146. Springer, 2006.

[TSG02] Elena Trichina, Domenico De Seta, and Lucia Germani. Simplified Adaptive Multiplicative Masking for AES. In Burton S Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 187–197. Springer, 2002.

[TV04a] Kris Tiri and Ingrid Verbauwhede. A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation. In *2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), 16-20 February 2004, Paris, France*, pages 246–251. IEEE Computer Society, 2004.

[TV04b] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *DATE*, 2004.

[vE85] Wim van Eck. Electromagnetic radiation from video display units: An eavesdropping risk? *Computers & Security*, 4(4):269–286, 1985.

[Wel47] Bernard L Welch. The generalization of Student's problem when several different population variances are involved. *Biometrika*, pages 28–35, 1947.

Part II

Publications

List of Publications

International journals

1. Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016.

   This is an extended version of the CHES 2015 publication, invited and accepted to the special issue of JCEN.

Lecture Notes in Computer Science

2. Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.
   – See p. 71.

3. Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Generic DPA attacks: Curse or blessing? In Emmanuel Prouff, editor, *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, volume 8622 of *Lecture Notes in Computer Science*, pages 98–111. Springer, 2014.

4. Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. A note on the use of margins to compare distinguishers. In *Constructive Side-Channel Analysis and Secure Design - 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers*, pages 1–8, 2014.

5. Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

6. Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, and Svetla Nikova. Higher-Order Glitch Resistant Implementation of the PRESENT S-Box. In Berna Ors and Bart Preneel, editors, *Cryptography and Information Security in the Balkans - First International Conference, BalkanCryptSec 2014, Istanbul, Turkey, October 16-17, 2014, Revised Selected Papers*, volume 9024 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2014.

7. Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the AES s-box. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.

8. Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1 GHz. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2015.
   – See p. 97.

9. Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015.
   – See p. 123.

10. Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.
    – See p. 151.

11. Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-lwe masking. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2016.

12. Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In Thomas Peyrin, editor, *Fast Software Encryption, 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016*, volume 0000 of *Lecture Notes in Computer Science*, page 20. Springer, 2016.
– See p. 183.

ACM

13. Junfeng Fan, Oscar Reparaz, Vladimir Rozic, and Ingrid Verbauwhede. Low-energy encryption for medical devices: security adds an extra design dimension. In *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pages 15:1–15:6. ACM, 2013.

Technical reports

14. Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. Compact and Side Channel Secure Discrete Gaussian Sampling. *IACR Cryptology ePrint Archive*, 2014:591, 2014.

Publication

Selecting time samples for multivariate DPA attacks

Publication Data

Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

Our contribution

Principal author.

Selecting Time Samples for Multivariate DPA Attacks

Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede

KU Leuven Dept. Electrical Engineering-ESAT/SCD-COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Masking on the algorithm level, i.e. concealing all sensitive intermediate values with random data, is a popular countermeasure against DPA attacks. A properly implemented masking scheme forces an attacker to apply a higher-order DPA attack. Such attacks are known to require a number of traces growing exponentially in the attack order, and computational power growing combinatorially in the number of time samples that have to be exploited jointly. We present a novel technique to identify such tuples of time samples before key recovery, in black-box conditions and using only known inputs (or outputs). Attempting key recovery only once the tuples have been identified can reduce the computational complexity of the overall attack substantially, e.g. from months to days. Experimental results based on power traces of a masked software implementation of the AES confirm the effectiveness of our method and show exemplary speed-ups.

Keywords: Time sample selection, multivariate side-channel attack, masking, reverse-engineering.

1 Introduction

Side-channel attacks are used to break implementations of cryptographic algorithms in embedded devices. Since the introduction by Kocher [Koc96] in the late nineties, they have been refined and a series of countermeasures have been designed to thwart them. A particularly popular countermeasure against Differential Power Analysis (DPA) attacks [KJJ99] is d-order masking [CJRR99, GP99], since it enjoys a formal proof of security against higher-order DPA attacks [CJRR99, Mes00] of order d or less. d-order masking is based on splitting every sensitive intermediate value in $d + 1$ shares and we consider the case that they are manipulated at distinct times, as is typical for software implementations. $d + 1$-order DPA and $d + 1$-variate Mutual Information Analysis (MIA) attacks [GBPV10, PR09] (from

now on referred to as multivariate attacks together) allow to break d-order masked implementations by analyzing tuples of $d+1$ time samples, corresponding to all shares of a masked sensitive variable, from each trace. However, multivariate attacks are significantly more difficult to mount than univariate attacks for two reasons. First, attacks exploiting higher-order moments are exponentially more sensitive to noise as the masking order d increases [CJRR99, SP06]. As a consequence, the number of traces required to mount a successful attack grows exponentially in d. Second, multivariate attacks need to search over $d+1$-tuples of time samples. The computational complexity of the attacks therefore grows combinatorially in the attack order $d+1$. Hence, secure implementations use a masking order d in combination with a suitable noise level to ensure that an attack will require a sufficiently large number of traces and a heavy amount of computation, such that the attack becomes impractical.

Related work.

Most related works on non-profiled multivariate attacks start from the assumption that the time samples where the shares of the targeted, masked sensitive variable leak are known, and focus on the key recovery [GBPV10, JPS05, Mes00, PR09, PRB09, SVO+10, WW04]. Few related works tackle the problem of identifying (tuples of) interesting time samples before key recovery, and they do so with heuristic approaches. Agrawal et al. [AARR02] describe a method to identify tuples of time samples that requires a chosen input adversarial model and that can only exploit the leakage of single bits. Their method is tailored to Boolean masking and the measurements can not be re-used for key recovery, due to the way the inputs are chosen. Oswald et al. [OMHT06] essentially propose an exhaustive search over all $d+1$-tuples of time samples in a small time window that is selected based on an *educated guess*. The interpretation of *educated guess* is left to the practitioner. Note that the guess does not select tuples of time samples, but a window of time samples that has to be searched for a tuple exhaustively in combination with key recovery. This method can be applied with known inputs or outputs and, in principle, to any masking scheme. The approach suggested by Lemke and Paar [LP07] and Gierlichs et al. [GBPV10] is to examine the empirical variance of several power traces when the input data is kept constant, i.e. it requires a chosen input adversarial model. In an ideal case, the variance is then caused only by masking, and therefore time samples with high variance mostly correspond to time samples where the masks or masked variables are being processed. Note that also this method does not identify tuples of time samples but a set of samples that has to be searched for a tuple exhaustively in combination with key recovery. The measurements can not be re-used for key recovery and, in principle, the method can be applied to any masking scheme.

In summary, the *educated guess* of Oswald et al. is the only method described in the literature that can be applied in black-box conditions and with known inputs or outputs.

Contribution.

We present a novel method for identifying interesting $d+1$-tuples of time samples before key recovery. It is not heuristic but systematic and ranks all possible $d+1$-tuples of time samples in a given window according to their dependency on, informally speaking, "typical attack targets". It does not provide a qualitative yes/no decision, but instead ranks tuples with respect to a meaningful metric such that there is a natural order in which to attack them. Our technique can lead to a substantial improvement in the computational efficiency of multivariate attacks compared to exhaustive search over the same window of time samples, since it retains only a small fraction of all possible $d+1$-tuples for key recovery. The relative improvement depends on the size of the subkeys that are attacked. In absolute terms, the improvement becomes more pronounced with increasing attack order $d+1$, increasing size of the time window, and increasing number of traces.

Our approach is based on mutual information and is fully generic: it applies to attacks of any order $d+1$, including univariate attacks against unmasked implementations, it applies to all possible masking schemes, it requires only a known input or output scenario, it can traverse S-boxes, locate shares of the masked S-box output, and it does not require any restrictive assumptions on the device leakage behavior. In other words, our method does not require more restrictive assumptions than a generic MIA attack [GBTP08].

Paper organization.

In Sect. 2 we introduce our notation, recall the basics of masking and discuss state-of-the-art multivariate attacks. In Sect. 3 we present our technique together with an analysis of how and why it works. We discuss its efficiency, impact, and possible refinements in Sect. 6. In Sect. 5 we present experimental results that validate our proposal and highlight some of its interesting properties. Section 6 concludes the paper.

2 Preliminaries

In this paper we consider only non-profiled, multivariate attacks. *Interesting* tuples are tuples of time samples that carry leakage of all shares of a masked variable that is a (possibly keyed) function of the plaintext.

2.1 Notation

Capital letters in bold face, e.g. $\mathbf{M}$, denote random variables. Lowercase letters, e.g. m, denote a specific value of $\mathbf{M}$, e.g. $\mathbf{M} = m$. $\mathbf{M}_i$ are mask bytes, $\mathbf{P}$ is a plaintext byte, $\mathbf{K}$ is a key byte, and `S-box` is a cryptographic S-box. $\mathbf{L}(t)$ is the random variable corresponding to the measured side-channel leakage at time t. $t_{\mathbf{M}}$ denotes the instant when the device is manipulating the random variable $\mathbf{M}$. $\mathbf{I}(\mathbf{A};\mathbf{B};\mathbf{C})$ denotes the multivariate mutual information between $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ [BGP+11, GBPV10] and is computed as

$$\mathbf{I}(\mathbf{A};\mathbf{B};\mathbf{C}) = \mathbf{I}(\mathbf{A};\mathbf{B}) - \mathbf{I}(\mathbf{A};\mathbf{B}|\mathbf{C})\,. \tag{1}$$

Note that, if $\mathbf{A}$ and $\mathbf{B}$ are independent, $\mathbf{I}(\mathbf{A};\mathbf{B}) = 0$ and $\mathbf{I}(\mathbf{A};\mathbf{B};\mathbf{C}) \leq 0$.

2.2 Masking

Masking was introduced by Goubin and Patarin [GP99] and by Chari et al. [CJRR99] (together with a proof of security) as a sound approach to protect implementations against first-order DPA attacks. In a d-order masked implementation, every sensitive variable $\mathbf{Z}$ is randomly split into $d+1$ shares $\mathbf{M}_1, \ldots, \mathbf{M}_d, \mathbf{V}$ satisfying

$$\mathbf{M}_1 \star \ldots \star \mathbf{M}_d \star \mathbf{V} = \mathbf{Z}\,, \tag{2}$$

where $\star$ is some suitable group operator.

The security of properly implemented masking schemes relies on the fact that even if the adversary manages to know any information about up to d shares out of $d+1$ (for example, via side-channel leakage), he cannot learn any information about the sensitive variable $\mathbf{Z}$.

Throughout the paper we assume that the shares $\mathbf{M}_1, \ldots, \mathbf{M}_d, \mathbf{V}$ are manipulated (and leak) separately at different time instants. Further, we assume that these time instants and the values of the shares are unknown to the adversary.

2.3 Multivariate attacks

Masked implementations of order d can in theory always be broken by $d+1$-variate attacks as originally proposed by Messerges [Mes00] and Chari et al. [CJRR99]. They exploit the statistical dependence between the leakage of the $d+1$ shares and the sensitive variable $\mathbf{Z}$. There are essentially two different methods for performing multivariate attacks.

The first approach [CJRR99, JPS05, Mes00, OMHT06, PRB09, WW04] consists in reducing the problem to a univariate scenario by preprocessing each trace, and then

running a first-order attack on the preprocessed traces. The preprocessing generates a new trace from all possible $d+1$-tuples of distinct time samples of the original trace, where for each tuple the $d+1$ time samples are combined with a so-called *combination function* (typically the absolute difference [Mes00] or the centered product [PRB09]). The second approach, proposed by Prouff and Rivain [PR09] and Gierlichs et al. [GBPV10], does not rely on a preprocessing step but directly uses multivariate MIA for the attack.

A major shortcoming of both methods is that they suffer from the effect known as "combinatorial explosion" and hence combinatorial time complexity in $d+1$. Both methods aim to recover subkeys while, at the same time, searching for a suitable $d+1$-tuple of time samples in the traces. In the first approach, the preprocessed traces are $\binom{L}{d+1}$ time samples long, where L is the trace length. These traces have to be processed for each hypothesis on the subkey. In the second approach, the distinguisher should be computed for each of the $\binom{L}{d+1}$ $d+1$-tuples, and for each hypothesis on the subkey.

Hence, it is very important to identify the interesting tuples (or to narrow down a window of time samples as much as possible) prior to key recovery in order to keep the computational complexity of a multivariate attack at a feasible level.

3 Identifying interesting tuples of time samples

In this section we explain how to identify interesting $d+1$-tuples of time samples prior to key recovery. Note that we focus our attention on this aspect and that key recovery is not the primary focus of the paper. For clarity in the exposition, in what follows we assume a first-order Boolean masking scheme (two shares) and a noise-free scenario. The practical results presented in Sect. 5 are based on measured power traces.

3.1 Core idea

Let us consider a scenario with fixed plaintext, fixed key, and sensitive intermediate value $\mathbf{Z} = F_k(p)$, where F_k is some key-dependent function (for example, $F_k(p) = \texttt{S-box}(p \oplus k)$). The key observation is that the mutual information between the leakages at time instants corresponding to the manipulation of the mask $\mathbf{M}_1$ and the masked intermediate value $\mathbf{V} = \mathbf{M}_1 \oplus F_k(p)$ is non-zero. That is,

$$\mathbf{I}(\mathbf{L}(t_{\mathbf{M}_1}); \mathbf{L}(t_{\mathbf{V}})) > 0\,. \tag{3}$$

The interpretation is straightforward: leakage at $t_{\mathbf{V}}$ depends only on $\mathbf{V}$, which varies in function of only the mask $\mathbf{M}_1$ (since the plaintext and the key are fixed), and some information about the mask is leaked at $t_{\mathbf{M}_1}$. Hence, the information shared

between leakage at $t_{\mathbf{M}_1}$ and $t_{\mathbf{V}}$ is non-zero. On the other hand, the information shared between leakage at two unrelated time samples t_0 and t_1 is zero

$$\mathbf{I}(\mathbf{L}(t_0); \mathbf{L}(t_1)) = 0 \tag{4}$$

because no relation exists between data handled at t_0 and at t_1. Thus, Eqs. (3) and (4) allow us to distinguish pairs of time samples that contain leakage of dependent variables (case of Eq. (3)) from those pairs that contain leakage of independent variables, that are irrelevant for the multivariate attack (case of Eq. (4)). Note that not all pairs of time samples that contain leakage of dependent variables carry some information about the key. For example, if the same mask is manipulated at t_0 and t_1, then $\mathbf{I}(\mathbf{L}(t_0); \mathbf{L}(t_1)) > 0$.

We stress that the value of $\mathbf{K}$ need not be known, and no hypothesis on it be made.

The general case.

In the above example we required a fixed plaintext and thus a chosen plaintext scenario. We can relax this assumption and instead work with known (varying) plaintexts. Suppose that the device is manipulating the plaintext byte $\mathbf{P}$, the mask $\mathbf{M}_1$ and the masked intermediate value $\mathbf{V}$ such that $\mathbf{V} = \mathbf{M}_1 \oplus F_k(\mathbf{P})$ at time instants $t_{\mathbf{P}}$, $t_{\mathbf{M}_1}$ and $t_{\mathbf{V}}$, respectively. The natural extension of the core observation to known varying plaintexts is that $\mathbf{L}(t_{\mathbf{P}})$, $\mathbf{L}(t_{\mathbf{M}_1})$ and $\mathbf{L}(t_{\mathbf{V}})$ are not independent, and therefore the mutual information between them is non-zero

$$\mathbf{I}(\mathbf{L}(t_{\mathbf{M}_1}); \mathbf{L}(t_{\mathbf{V}}); \mathbf{L}(t_{\mathbf{P}})) \neq 0\,. \tag{5}$$

At three unrelated time samples t_0, t_1 and t_2, on the other hand, the mutual information is zero

$$\mathbf{I}(\mathbf{L}(t_0); \mathbf{L}(t_1); \mathbf{L}(t_2)) = 0\,. \tag{6}$$

The interpretation follows the same lines as in the particular case. Leakage at $t_{\mathbf{V}}$ depends only on $\mathbf{V}$, which now varies in function of the plaintext and the mask (since the key is fixed), and some information about the mask and the plaintext is leaked at $t_{\mathbf{M}_1}$ and $t_{\mathbf{P}}$, respectively. Thus, the information shared between $\mathbf{L}(t_{\mathbf{M}_1})$, $\mathbf{L}(t_{\mathbf{V}})$ and $\mathbf{L}(t_{\mathbf{P}})$ is non-zero.

Note that it is not necessary to search for $t_{\mathbf{P}}$, nor is it necessary for $t_{\mathbf{P}}$ to physically exist in the power traces. By assumption, the plaintext is known, so it is possible to substitute $\mathbf{L}(t_{\mathbf{P}})$ with the leakage of the known plaintext under some hypothesized leakage model $\tilde{\mathbf{L}}(\mathbf{P})$. This makes the analysis faster since one has to search only for a pair of time instants ($t_{\mathbf{M}_1}$ and $t_{\mathbf{V}}$) instead of searching for a triplet. The choice of $\tilde{\mathbf{L}}$ will be discussed in Sect. 3.3.

We can hence use Eqs. (5) and (6) to distinguish dependent triplets from independent triplets. In addition, and contrary to the particular case with fixed

plaintext, all identified tuples are now interesting tuples and all relate to the *specific* plaintext byte $\mathbf{P}$. Most of them carry some information about the key and can be useful for a key recovery attack. The only possible type of tuple that will be identified as interesting although it does not carry some information about the key is the one corresponding to all shares of the specific, masked plaintext byte. We discuss this in more detail in Sect. 3.3.

3.2 Suggested workflow for multivariate attacks

The previous observations allow an attacker to identify interesting $d+1$-tuples of time samples prior to key recovery. Again, we use $d = 1$ in the explanation. The proposed workflow divides an attack in three phases:

Step 1. (Window selection) The adversary uses any available mean to narrow down the time window to analyze. For example, the adversary could select a small window based on an *educated guess* [OMHT06], if possible. Obviously, care has to be taken to not discard too many time samples since the window must contain at least one interesting $d+1$-tuple.

Step 2. (Tuple selection) The adversary estimates $\mathbf{I}(\mathbf{L}(t_1); \mathbf{L}(t_2); \tilde{\mathbf{L}}(\mathbf{P}))$ for all (t_1, t_2) with $t_1 > t_2$ in the remaining window, and keeps a list of pairs of time samples yielding negative mutual information with large absolute value.

Step 3. (Key recovery attack) The adversary performs the preferred strategy for a bivariate attack on traces consisting only of the pairs of time samples in the list. These traces consist of a few pairs of time samples, and hence the key recovery step is much faster.

3.3 Which tuples of time samples pop up?

The adversary has freedom to choose the hypothesized leakage model $\tilde{\mathbf{L}}$ for the plaintext. Depending on the choice of $\tilde{\mathbf{L}}$, different tuples of time samples will be identified. In this section we analyze two cases.

$\tilde{\mathbf{L}}$ is the identity function.

When the adversary computes the mutual information between time samples and a plaintext byte, i.e. $\tilde{\mathbf{L}}(\mathbf{P}) = \mathbf{P}$, he will be able to identify all tuples corresponding to all shares of any (sensitive) variable of the form $\mathbf{Z} = F_k(\mathbf{P})$. In particular, the method is able to identify the shares $(\mathbf{M}_1, \mathbf{V})$ with $\mathbf{V} = \mathbf{P} \oplus \mathbf{M}_1$, $\mathbf{V} = \mathbf{P} \oplus \mathbf{K} \oplus \mathbf{M}_1$ and $\mathbf{V} = \texttt{S-box}(\mathbf{P} \oplus \mathbf{K}) \oplus \mathbf{M}_1$, since the key is fixed.

This result is useful, as it allows the attacker to locate both the masked variables *before* the S-box (masked plaintext and masked S-box input) as well as the masked variables *after* the S-box (masked S-box output). Note that it is irrelevant if the masks before and after the S-box are the same. If the mask does not change, the identified tuples of time samples will share one component.

$\tilde{\mathbf{L}}$ is an approximation of the device leakage behavior.

If the attacker chooses $\tilde{\mathbf{L}}$ as an approximation of the leakage behavior $\mathbf{L}$, he will be able to identify all tuples of time samples corresponding to all shares of any (sensitive) variable of the form $\mathbf{Z} = F_k(\mathbf{P})$ appearing *before* the S-box (e.g. masked plaintext and masked S-box input). For a typical S-box, he will not be able to identify tuples of time samples corresponding to shares of sensitive variables *after* the S-box. The intuitive reasoning behind this is that knowledge of the distribution of the plaintext's *leakage* does not give sufficient information for guessing the distribution of the S-box output's leakage. The advantage of this choice, compared to the identity function, is the ease of estimation, see Sect. 4.1. Disadvantages are that one cannot locate shares of masked variables after the S-box and that one relies on an assumption about the device leakage behavior.

Note that we compute the mutual information according to Eq. (1), and not as

$$\begin{aligned}\mathbf{I}((\mathbf{L}(t_0), \mathbf{L}(t_1)); \tilde{\mathbf{L}}(\mathbf{P})) = \\ \mathbf{I}(\mathbf{L}(t_0); \tilde{\mathbf{L}}(\mathbf{P})) + \mathbf{I}(\mathbf{L}(t_1); \tilde{\mathbf{L}}(\mathbf{P})) - \mathbf{I}(\mathbf{L}(t_0); \mathbf{L}(t_1); \tilde{\mathbf{L}}(\mathbf{P}))\,,\end{aligned} \tag{7}$$

where the last of the three terms is in turn given by Eq. (1) [BGP+11]. The reasoning for this choice is straightforward. The first two terms of Eq. (7) capture first-order leakage of variables that depend on $\tilde{\mathbf{L}}(\mathbf{P})$, e.g. unmasked plaintext, unmasked S-box input and, depending on the choice of $\tilde{\mathbf{L}}$, unmasked S-box output. By assumption, the masking scheme is properly implemented and there is no first-order leakage of sensitive variables. Hence, the only first-order leakage that these terms could capture is that of the unmasked plaintext, which is of no use for our purpose. By omitting the two terms and using Eq. (1) we ensure that only interesting tuples yield non-zero mutual information.

Moreover, Eq. (1) allows us to target very specific tuples. For our interesting tuples it holds that $\mathbf{I}(\mathbf{L}(t_{\mathbf{M}_1}), \mathbf{L}(t_{\mathbf{V}})) = 0$ such that interesting tuples yield strictly negative mutual information, see (1).

4 Discussion

In this section we discuss several aspects of the proposed workflow for multivariate attacks, such as its efficiency, refinements and additional applications.

4.1 Efficiency analysis

We evaluate the efficiency of the proposed workflow with respect to the running time and the number of traces needed, and we compare these numbers to those of a "classical" multivariate MIA attack that uses exhaustive search instead of step 2. Although the proposed method is not limited to a particular multivariate attack technique for step 3, using multivariate MIA here allows us to draw important conclusions regarding the efficiency of the proposed workflow, since the numbers can be directly compared. In both cases we focus the attacks on the (masked) S-box output. According to the previous section, this choice implies that step 2 of the proposed workflow uses the identity function $\tilde{\mathbf{L}}(\mathbf{P}) = \mathbf{P}$. We analyze two different scenarios:

(a) Unknown leakage behavior. Step 3 of the proposed workflow and the "classical" MIA both use the identity leakage model, or possibly some truncated identity leakage model in case of a bijective S-box. The point here is that both step 3 and the "classical" MIA use the same leakage model.

(b) Known leakage behavior $\mathbf{L}$ equal to Hamming weight leakage. Step 3 of the proposed workflow and the "classical" MIA both use the Hamming weight leakage model.

Running time.

We assume that after step 1 the traces are L time samples long and contain at least one tuple of time samples corresponding to all shares of the masked S-box output. We further assume that all attacks are provided with sufficiently many traces, i.e. there are no PDF estimation problems.

In scenario (a) the running time of the "classical" MIA attack is given by $\binom{L}{d+1} \times \alpha \times |K|$, where $\binom{L}{d+1}$ is the number of $d+1$-tuples of time samples to analyze, α is the time it takes to compute the MIA distinguisher for one $d+1$-tuple of time samples and one subkey hypothesis using the identity leakage model, and $|K|$ is the number of subkey hypotheses. In scenario (b) the running time of the "classical" MIA attack is $\binom{L}{d+1} \times \beta \times |K|$, where β is the time it takes to compute the MIA distinguisher for one $d+1$-tuple of time samples and one subkey hypothesis using the Hamming weight leakage model.

Table 1 – Running time of MIA attacks using the proposed and the "classical" workflow.

	Proposed workflow	"Classical" MIA	Improvement factor
Scenario (a)	$\binom{L}{d+1} \times \alpha + \lvert K\rvert \times \alpha \times \gamma$ $\approx \binom{L}{d+1} \times \alpha$	$\binom{L}{d+1} \times \alpha \times \lvert K\rvert$	$\approx \lvert K\rvert$
Scenario (b)	$\binom{L}{d+1} \times \alpha + \lvert K\rvert \times \beta \times \gamma$ $\approx \binom{L}{d+1} \times \alpha$	$\binom{L}{d+1} \times \beta \times \lvert K\rvert$	$\approx \lvert K\rvert \times \beta/\alpha$

In scenario (a) the running time of step 2 of the proposed workflow is given by $\binom{L}{d+1} \times \alpha$, and the running time of step 3 is $|K| \times \alpha \times \gamma$, where γ is the number of $d+1$-tuples in the list of interesting tuples generated in step 2. We have that $\gamma \geq 1$ and typically γ is much smaller than L. The combined running time of steps 2 and 3 is $\binom{L}{d+1} \times \alpha + |K| \times \alpha \times \gamma$. In scenario (b) the running time of step 2 is again $\binom{L}{d+1} \times \alpha$ and the running time of step 3 is $|K| \times \beta \times \gamma$. The combined running time of both steps is $\binom{L}{d+1} \times \alpha + |K| \times \beta \times \gamma$. Note that in both scenarios (a) and (b), the total running time of the proposed workflow is dominated by step 2. Table 1 summarizes these numbers and shows that the proposed workflow essentially runs $|K|$ times faster.

So far we have limited this analysis to attacks against a single subkey. For attacking multiple subkeys, it may be that only recovering the first subkey is hard and that the interesting tuples of time samples related to the other subkeys can be easily guessed once the tuple related to the first subkey has been found. But it may also be that recovering the other subkeys requires basically the same computation as recovering the first subkey. In either case, the improvement factor is essentially $|K|$. In the latter case, this improvement applies to recovering *each* subkey, which is not obvious since we express the improvement as a factor. Further, we note that the improvement factor is independent of the masking order d and the window size L. However, in absolute terms the running time improvement increases substantially with increasing attack order $d+1$, L and the number of traces. Finally, we point out that the analysis holds independently of the method used to estimate the mutual information, as long as we assume that all involved estimations of mutual information use the same method.

Number of traces needed.

It is not straightforward to make a precise but general statement about the number of traces needed for our method to successfully locate interesting tuples. Many factors play a role. We make a brief assessment and describe two of the effects that have to be considered.

First, we consider an idealized scenario where steps 2 and 3 succeed as soon as the same precision for the estimations is achieved. In this case, in scenario (b) (Hamming weight leakage model), step 2 may require more traces to pinpoint the interesting $d+1$ tuples of time samples than step 3 to recover the key. This is due to the fact that, in the attack step, the estimation of $\mathbf{I}(\mathbf{L}(t_{\mathbf{V}}); \mathbf{L}(t_{\mathbf{M}}); \mathrm{HW}(\mathbf{Z}))$ with $\mathbf{Z} = \texttt{S-Box}(\mathbf{P} \oplus k)$ for a hypothesized k requires generally less traces than an equally precise estimation of $\mathbf{I}(\mathbf{L}(t_{\mathbf{V}}); \mathbf{L}(t_{\mathbf{M}}); \mathbf{P})$ in the tuple selection step. This is because of the different number of classes for $\mathrm{HW}(\mathbf{Z})$ and for $\mathbf{P}$. In the case of AES, there are 256 different possible values for $\mathbf{P}$, while there are only nine different possible values for $\mathrm{HW}(\mathbf{Z})$. Nevertheless, since step 2 requires a larger number of traces, these traces must be obtained and may be used in step 3. A "classical" $d+1$-variate MIA attack requires the same (smaller) number of traces as step 3.

In scenario (a) ((possibly truncated) identity leakage model) the previous effect is typically less pronounced and thus the difference in the number of traces required in each step is smaller. The same holds for the difference in the number of traces needed for step 2 and a "classical" $d+1$-variate MIA attack.

Second, the precision of the mutual information estimates required in step 3 to distinguish the correct key hypothesis from incorrect ones may not be the same as the precision required to distinguish an interesting tuple from a non-interesting one in step 2. The relation between these precisions can be almost arbitrary. However, it should typically hold that the precision required by an attack against the S-box output in step 3 is not higher than the precision required in step 2.

Summarizing, in scenario (a) the proposed workflow offers a running time improvement factor in the order of magnitude of $|K|$, possibly at the cost of an increased number of traces. In scenario (b) the proposed workflow requires more traces than a "classical" attack but still offers an interesting running time improvement factor. It offers a trade off. Whether the trade off is attractive depends on the ratio β/α in the running time improvement factor, and on how many more traces are required.

4.2 On the S-box

The fact that the method can distinguish all $d+1$ tuples corresponding to all shares of any (sensitive) variable of the form $\mathbf{Z} = F_k(\mathbf{P})$ can be used to traverse bijective S-boxes without making any hypothesis on the subkey. This is because the S-box input is a keyed permutation of the plaintext, and the S-box output is a permutation of the S-box input. Both permutations are transparent to mutual information when using the identity function $\tilde{\mathbf{L}}(\mathbf{P}) = \mathbf{P}$.

It is less obvious, nevertheless true, that the method also works in the case of non-injective S-boxes, as for instance in DES. The reasoning is similar to the above.

The S-box input is a keyed permutation of the plaintext. The S-box output is not a permutation of the S-box input, but a non-injective function of it. Therefore, if we use the identity function and condition on the plaintext, the S-box can be traversed just like a bijective S-box, and interesting tuples of time samples after the S-box can be identified. Note that a non-injective S-box cannot be traversed from output to input in the same way.

4.3 Additional applications

The method described in this paper is fully generic and does not place any restrictive assumption on the specific targeted implementation. However, the method benefits from the available specificities of the implementation. For example, an adversary could mount the following strategy if he knows that the device's leakage behavior is close to the Hamming weight model. Using the Hamming weight model, the adversary first locates tuples corresponding to the S-box input to narrow down the time window. Then, using the identity function, he searches in that window for the S-box output.

The adversary could also locate tuples corresponding to the S-box input of the next S-box lookup to further narrow down the time window.

Bit-tracing [JO05] is a technique used to track the time instants when a predictable variable is handled in the execution flow of an unknown implementation. This is a useful technique to reverse-engineer unknown implementations. The ideas in Section 3.1 can be exploited to track masked variables during the execution of an algorithm. Note that the fact that the proposed method can traverse S-boxes (by the arguments given in Sect. 4.2) can also lead to a significant speed-up in the bit-tracing process of masked implementations.

4.4 Estimation of mutual information

We note that any suitable method for estimating the mutual information or the required probability distributions, e.g. histograms [GBTP08], kernel density estimation [PR09], B-splines [Ven10], statistical moments [LB10], parametric methods [PR09], and any similar metric, e.g. Kullback-Leibler divergence [VS09], Kolmogorov-Smirnov test [VS09], Cramér-von-Mises test [VS09], can be used.

Available knowledge about the device leakage behavior, e.g. close to the Hamming weight model, can be used to speed up the estimations. Here we do not refer to the choice of $\tilde{\mathbf{L}}$ but to the leakage variables.

4.5 Key recovery step

By construction, our method identifies tuples of time samples that correspond to all shares of a masked (sensitive) variable. It does so irrespective of the particular dependencies between each share and its side-channel leakage. Therefore, a generic multivariate MIA attack with (possibly truncated) identity leakage model appears to be most suited to exploit the unknown dependencies, in general. However, if standard assumptions approximate the leakage behavior good enough or the specific leakage behavior is known, the identified tuples can be exploited more efficiently with adapted multivariate MIA or higher-order DPA attacks.

The proposed method can identify interesting tuples that relate to a specific plaintext byte, but it cannot *per se* focus on interesting tuples that correspond to a *specific* function of that plaintext byte. As a consequence, the method will in general not discriminate between interesting tuples that correspond to all shares of the masked plaintext, the masked S-box input or the masked S-box output. Clearly, the latter is preferable for an attack. In our experiments we noted that enough interesting tuples corresponding to all shares of the masked S-box output appeared at the top of the ranked list.

5 Experiments

In this section we present experimental results of our method, insight on its computation and a performance evaluation. We note that all "numbers of traces" reported in this section cannot be generalized to other platforms and implementations.

5.1 Measurements

We use an 8-bit microcontroller of Atmel's AVR family in a smart card plastic body as platform for our experiments. The microcontroller runs a first-order Boolean masked implementation of AES-128 encryption that follows the lines of [HOM06]. This concrete implementation uses six independent mask bytes for one encryption. Before the SubBytes operation, all state bytes are protected by the same mask $\mathbf{M}_0$. After the SubBytes operation, all state bytes are protected by the same mask $\mathbf{M}_1$. Before MixColumns, each column of the state is remasked with $\mathbf{M}_2, \ldots, \mathbf{M}_5$. After MixColumns, each column of the state is masked with $\mathbf{M}'_2, \ldots, \mathbf{M}'_5$ that depend on $\mathbf{M}_2, \ldots, \mathbf{M}_5$. Shiftrows does not affect the masking and after the next AddRoundKey operation, all state bytes are again protected by $\mathbf{M}_0$ due to the masked key schedule. Note that the six masks are re-used to protect all rounds. There are no additional countermeasures.

We obtained 50 000 power traces from encryptions of randomly chosen plaintexts with a fixed key and random masks. The card was clocked at 4MHz and we used a sampling frequency of 200MS/s.

5.2 Selection of a time window: step 1

To reduce the computational burden, we restricted the measurements to cover only the first 1.5 rounds of the encryption. This was done based on an educated guess on the SPA features present in the power traces. Then, we compressed the traces by integration to one point per clock cycle. As a result, each compressed trace comprises 800 points. The subsequent analyses were carried out on these compressed traces.

5.3 Computation of the method: step 2

To show the full potential of the method, we chose $\tilde{\mathbf{L}}$ to be the identity function. In what follows, $\mathbf{P}$ refers to the third plaintext byte, an arbitrary choice. We estimate densities with histograms (using nine bins for each dimension unless otherwise stated, because we expect a leakage behavior close to the Hamming weight/distance model) and we use $\hat{}$ to indicate estimates, e.g. $\hat{\mathbf{I}}$ is an estimate of $\mathbf{I}$. The computation of step 2 is split into two terms:

$$\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1);\mathbf{P}) = \hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)) - \hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{P})\,. \tag{8}$$

In our experiments, we noted that a straightforward computation of this expression can result in inconvenient estimation errors. The reason lies in the different number of traces used to estimate each term on the right side of Eq. (8). $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1))$ is computed with all available traces, say T. The second term is computed as

$$\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{P}) = \sum_{p=0}^{255} \hat{\Pr}(\mathbf{P}=p)\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{P}=p) \tag{9}$$

and for the computation of each summand about $T/256$ traces are used. This difference in the number of traces translates into different estimation accuracies for each term in Eq. (8), burying the small relevant difference between them due to the effect of $\mathbf{P}$ in the larger difference due to the different estimation accuracies.

To amend this, since we have that $\mathbf{I}(\mathbf{L}(t_0);\mathbf{L}(t_1)) = \mathbf{I}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{D})$ for a uniformly distributed *dummy* random variable $\mathbf{D}$ that is independent of the leakages and taking values in $\{0,\ldots,255\}$, we can compute $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1))$ in a way that resembles Eq. (9) and approximate it by

$$\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{D}) = \sum_{d=0}^{255} \hat{\Pr}(\mathbf{D}=d)\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{D}=d)\,. \tag{10}$$

This leads to equally (in-)accurate estimates for both terms in Eq. (8) and the difference between them is mostly due to the effect of $\mathbf{P}$.

To illustrate the effectiveness of step 2 we compute $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{D})$ and $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1)|\mathbf{P})$ from 50 000 measurements using the same bin distributions for both terms. We use this relatively large number of traces to present aesthetically pleasant figures. Far less traces are sufficient for the method to work.

Figure 1 (left) shows a plot of the values of the first term of Eq. (8), i.e. $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1))$ computed as Eq. (10), for $t_0, t_1 \in \{1,\ldots,800\}$ and $t_0 \neq t_1$. It is obviously sufficient to compute the values only for $t_0 < t_1$ or $t_0 > t_1$. The x- and y-axes both denote time. We plot a mean trace next to each of them for orientation. The values of mutual information are represented by different colors according to the color bar on the left side. We blank out most pairs of time samples, those that yield small values of mutual information, by plotting them in white. All pairs that yield mutual information values above a certain threshold are plotted in black.

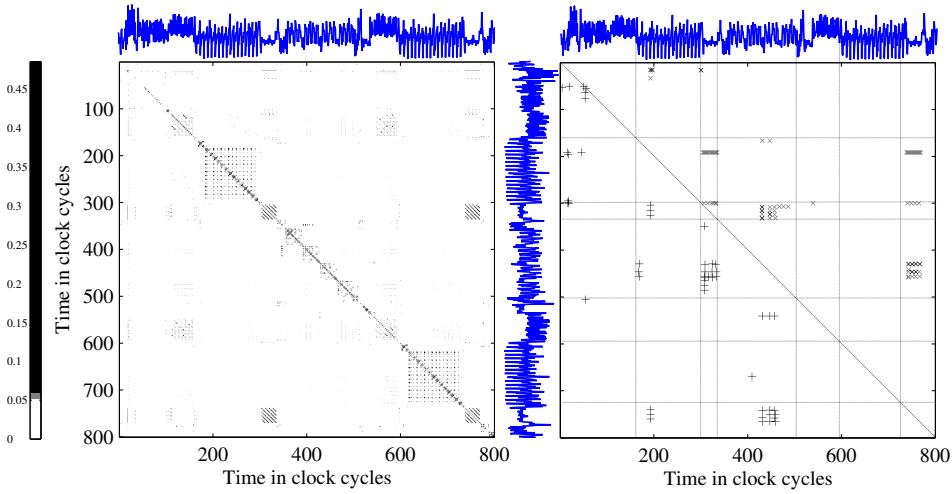


Figure 1 – Left: Matrix of $\hat{\mathbf{I}}(\mathbf{L}(t_0);\mathbf{L}(t_1))$ values. The color bar is in units of bits. A mean trace is plotted next to the axes. Right: Above diagonal, 'x': 100 pairs of time samples where a multivariate MIA attack succeeds. Below diagonal, '+': 100 top ranked pairs in the list of step 2.

We can see that the locations of the pairs have a clear structure and could possibly aid reverse-engineering of the implementation. Since we know the implementation,

we can easily relate parts of the figure to operations: AddRoundKey (approx. index 100 to 150), SubBytes (approx. index 200 to 300), remasking (approx. 300 to 350), four parts of MixColumn (approx. index 350 to 500), AddRoundKey (approx. index 550 to 600), followed by SubBytes and remasking in round two. These pairs are, however, not yet interesting pairs because it is not clear if they can be exploited by an attack (see the discussion of Eqs. (3) and (4)).

Next, we rank the list of pairs according to the result of Eq. (8). The 100 top ranked pairs in the list, i.e. negative mutual information and large absolute value, are depicted in the lower triangle of Figure 1 (right) with '+' symbols.

For the sake of comparison, we include in the upper triangle of the figure the 100 pairs of time samples where a multivariate MIA attack on the third key byte (using the Hamming weight leakage model on predicted S-box output values and 50 000 traces) achieves the largest nearest-rival distinguishing score [WO11], marked with 'x' symbols.

The partial match between the upper and the lower triangular matrix serves as a first visual evidence for the effectiveness of the method. In particular, the method is able to identify pairs corresponding to both shares of the S-box output of a specific state byte (here the third) without making any hypothesis about the key.

5.4 Performance evaluation of step 2

This section details the performance of the proposed method in finding the pairs that can be exploited for key recovery. Informally, we aim to decouple the performance of the proposed method from the performance of the key recovery attack itself, which is not the focus of this paper. To do so, we first define a set of *good* pairs of time samples that can be attacked and then we analyze the performance of the method in identifying *good* pairs among all possible pairs.

More precisely, we define sets of good pairs by running an attack on all pairs using 50 000 measurements and retaining the 100 resp. 290 pairs that lead to key recovery and have highest nearest-rival distinguishing score. Our choice for the size of the sets is somewhat arbitrary. The idea is simply to define one smaller set of very good pairs and a larger set that contains additional good pairs with lower nearest-rival distinguishing score. Since different attacks may favor different pairs, we define such sets for three cases: multivariate MIA on the S-box output, Correlation Power Analysis (CPA) [BCO04] with centered product combination function [PRB09] on the S-box output and CPA with same combination function on the S-box input (all using the Hamming weight leakage model). In total, we hence define six sets of good pairs.

Once the sets of good pairs are defined, we run step 2 parametrized by the number of traces. For each number of traces, we repeat the run of step 2 on 100 randomly

chosen sets of traces and, each time, keep the position of the best ranked good pair in the list generated by step 2. In other words, we test the pairs in descending order of their ranking (rank 1 is best) and stop as soon as a pair is good. This ranking position is the minimum size of the list from step 2 required for step 3 to succeed in that particular run for a given attack technique. Recall that, by definition, an attack on a good pair succeeds with a comfortable nearest-rival distinguishing score (albeit the absolute margin for a CPA attack on the S-box input is a lot smaller). We hence evaluate only the performance of step 2.

The distributions of the ranks of the best ranked good pairs are shown as boxplots in Fig. 2 (sets of 100 good pairs) and in Fig. 3 (sets of 290 good pairs). For both figures, the used attack techniques are, from left to right: MIA S-box output, CPA S-box output and CPA S-box input.

In the boxplots, the central mark is the median (2^{nd} quartile) and the box edges (solid) represent the 1^{st} and the 3^{rd} quartile. The whiskers (dashed) extend to $q_3 + 1.5(q_3 - q_1)$ and $q_1 - 1.5(q_3 - q_1)$, where q_1 and q_3 are the 1^{st} and 3^{rd} quartiles, respectively. Outliers are marked with '+' symbols.

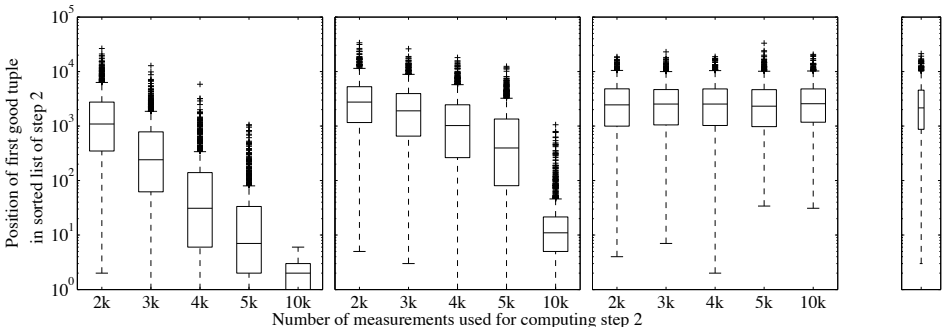


Figure 2 – Distribution of the ranking of the first *good* pair in the list of step 2. Left to right: MIA S-box output, CPA S-box output, CPA S-box input, hypothetical random method. 100 good pairs.

For comparison, the rightmost boxplot in each figure shows the distribution that a hypothetical method that ranks the pairs at random, instead of step 2, would produce. These distributions are independent of an attack technique and only relate to the number of good pairs among all pairs, here 100 resp. 290 out ouf $800 \times 799/2 = 319\,600$.

One can observe that the proposed method begins to identify good pairs (i.e. to perform better than a random guess) that are exploitable by multivariate MIA or CPA attacks on the S-box output when 3 000 traces or more are available. As the number of traces increases, the medians of the distributions become smaller, i.e. good pairs move steadily toward the top of the list.

One can also observe that our method ranks good pairs for multivariate MIA slightly higher than good pairs for CPA on the S-box output. On the other hand, the method is not able to identify good pairs for a CPA attack on the S-box input better than a random guess. We note that both behaviors are not a property of our method but probably related to our test platform and the implementation.

In the case of larger lists of 290 good pairs, the previously made observations mostly hold. As expected, the medians of the distributions are smaller than in the case of 100 good pairs, simply because even a random guess becomes more likely to succeed. In addition, we can observe that the method now ranks good pairs for multivariate MIA and CPA on the S-box output almost equally well.

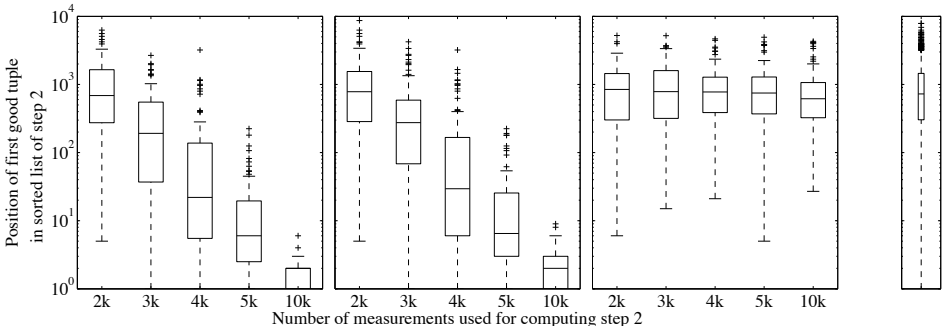


Figure 3 – Distribution of the ranking of the first *good* pair in the list of step 2. Left to right: MIA S-box output, CPA S-box output, CPA S-box input, hypothetical random method. S sets containing 290 pairs.

5.5 Practical attacks

The above results highlight important properties of our method and demonstrate that it is effective. In practice, one is however less interested in the exact rank of the first good pair in the sorted list, and more interested in the success rate of an attack end-to-end. This clearly involves the performance of our method *and* the efficiency of the attack used in step 3.

Table 2 shows success rates for steps 2 and 3 together. First we use a given number of randomly chosen traces to compute step 2. Then we attack the $\gamma = 10$ resp. 100 best ranked pairs with multivariate MIA, CPA on the S-box output and CPA on the S-box input (as described before) in step 3, using the same traces. We repeat this procedure 100 times. For the numbers in the first row of the table, we considered an attack successful if the correct key leads to the smallest correlation (or mutual information) value (negative sign and highest absolute value), over all

Table 2 – Success rates for steps 2 and 3 together, for several parameters: number of traces, size γ of the list of step 2, key recovery attack.

Number of traces		2k	3k	4k	5k	10k		2k	3k	4k	5k	10k
MIA S-box output	$\gamma = 100$	3	15	59	83	100	$\gamma = 10$	0	3	34	53	100
CPA S-box output		11	41	75	95	100		1	11	48	66	100
CPA S-box input		1	0	0	1	0		2	0	0	1	0
MIA S-box output	$\gamma = 10$	0	2	15	35	100	$\gamma = 10$	0	0	7	17	90
CPA S-box output	factor 1.5	0	3	28	52	98	factor 2	0	0	10	17	78
CPA S-box input		0	0	0	0	0		0	0	0	0	0

evaluated γ pairs. For the numbers in the second row of the table, we additionally required the correct key to stand out at least by a factor of 1.5 compared to the nearest rival (left) and by a factor of at least 2 (right).

A first observation is that a CPA attack on the S-box input does not work in our concrete scenarios. CPA attacks on the S-box output converge slightly faster toward 100% success rate than multivariate MIA attacks on the S-box output. We can further see that, given enough traces, both attacks in step 3 eventually reach 100% success, even if we attack only the top ten pairs of step 2 and require the correct key to stand out by a factor of at least 1.5. These results confirm that the combination of steps 2 and 3 works in practice, and that step 2 is able to identify exploitable pairs of time samples. Interestingly, one can further see that multivariate MIA attacks on the S-box output have a small advantage over CPA attacks on the S-box output, if we require the correct key to stand out by a factor of at least 2.

5.6 Computational efficiency

In Tab. 3 we present empirical execution times for our implementations of the proposed workflow (steps 2 and 3) and the strategy that uses exhaustive search instead of step 2. Step 3 of the proposed workflow was performed with multivariate MIA on the S-box output (using the Hamming weight leakage model and list size $\gamma = 100$). For the exhaustive search strategy we evaluated two variants: multivariate MIA on the S-box output (using the Hamming weight leakage model) and CPA on the S-box output (with centered product preprocessing). All implementations were executed on the same processor on a single core. We note that the absolute execution times are heavily implementation-dependent and thus relative speed-ups are more interesting, since they are less tied to the particular implementation used.

A first observation regarding Tab. 3 is the speed-up achieved by the proposed workflow, compared to exhaustive search, when multivariate MIA is used for key

Table 3 – Empirical execution times for steps 2 and 3 ($\gamma = 100$) of the proposed workflow and several attacks using exhaustive search.

Number of traces	step 2 + step 3	Exhaustive search		Improvement factor
5 000	2m30s + 2s	MIA-HW	1h48m	43
		CPA	2h 48m	68
50 000	10m24s + 19s	MIA-HW	17h18m	97
		CPA	23h32m	132

recovery. This is a directly interpretable result that corresponds to scenario (b) in Sect. 4.1. The improvement factor in this case is of 43 when 5 000 and 97 when 50 000 traces are used, respectively. We observe that, for our implementations, the factor β/α depends on the number of traces.

One can further see that, for our implementations, applying the proposed workflow is even advantageous if exhaustive search is done with CPA. It achieves an improvement factor of 68 in the running time of the attack when 5 000 traces are used, and an improvement factor of 132 when 50 000 traces are used. However, we stress that this result is not universally valid. The speed-ups are heavily affected by the relative efficiency of our implementations of linear correlation and mutual information estimation.

As a final observation concerning Tab. 3, we remark the validity of the approximation we made in Tab. 1: the running time of the proposed workflow is dominated by step 2. Step 3 contributes at most 3% to the total running time if the list size is $\gamma = 100$.

6 Conclusion

Multivariate DPA attacks can suffer from the effect known as “combinatorial explosion” and hence combinatorial time complexity in the number of time samples that have to be exploited jointly. We presented a novel technique to identify such interesting tuples of time samples before key recovery. Compared to previous work on this topic, our method is not heuristic but systematic and works in black-box conditions using only known inputs (or outputs). Our technique can lead to a substantial improvement in the computational efficiency of multivariate attacks compared to exhaustive search over the same window of time samples, since it retains only a small fraction of all possible tuples for key recovery. Our approach is based on mutual information and is fully generic, i.e. it does not require more restrictive assumptions than a generic MIA attack. Experimental results based on power traces of a masked software implementation of the AES confirm the

effectiveness of the technique, highlight some of its interesting properties and attest attractive running time improvements. An aspect that is not fully explored in this paper and left for future work is a thorough analysis of the number of traces needed for the technique to work.

Acknowledgments

We thank the anonymous reviewers for their thorough evaluation and insightful comments.

This work was supported in part by the Research Council of KU Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II, by the Flemish Government FWO G.0550.12N and by the Hercules Foundation AKUL/11/19. Benedikt Gierlichs is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BGP+11] Lejla Batina, Benedikt Gierlichs, Emmanuel Prouff, Matthieu Rivain, François-Xavier Standaert, and Nicolas Veyrat-Charvillon. Mutual information analysis: a comprehensive study. *J. Cryptology*, 24(2):269–291, 2011.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In

Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[GBPV10] Benedikt Gierlichs, Lejla Batina, Bart Preneel, and Ingrid Verbauwhede. Revisiting higher-order DPA attacks:. In *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, pages 221–234, 2010.

[GBTP08] Benedikt Gierlichs, Lejla Batina, Pim Tuyls, and Bart Preneel. Mutual information analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 426–442, 2008.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, pages 239–252, 2006.

[JO05] Marc Joye and Francis Olivier. Side-channel analysis. In *Encyclopedia of Cryptography and Security*. 2005.

[JPS05] Marc Joye, Pascal Paillier, and Berry Schoenmakers. On second-order differential power analysis. In Josyula R Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005, 7th International Workshop, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, volume 3659 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in*

Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[LB10] Thanh-Ha Le and Maël Berthier. Mutual information analysis under the view of higher-order statistics. In *Advances in Information and Computer Security - 5th International Workshop on Security, IWSEC 2010, Kobe, Japan, November 22-24, 2010. Proceedings*, pages 285–300, 2010.

[LP07] Kerstin Lemke-Rust and Christof Paar. Gaussian mixture models for higher-order side channel analysis. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 14–27. Springer, 2007.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç K Koç and C Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *LNCS*, pages 238–251. Springer, 2000.

[OMHT06] Elisabeth Oswald, Stefan Mangard, Christoph Herbst, and Stefan Tillich. Practical second-order DPA attacks for masked smart card implementations of block ciphers. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 192–207. Springer, 2006.

[PR09] Emmanuel Prouff and Matthieu Rivain. Theoretical and practical aspects of mutual information based side channel analysis. In *Applied Cryptography and Network Security, 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings*, pages 499–518, 2009.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[SVO+10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 112–129, 2010.

[Ven10] Alexandre Venelli. Efficient Entropy Estimation for Mutual Information Analysis Using B-Splines. In Pierangela Samarati, Michael Tunstall, Joachim Posegga, Konstantinos Markantonakis, and Damien Sauveron, editors, *WISTP*, volume 6033 of *LNCS*, pages 17–30. Springer, 2010.

[VS09] Nicolas Veyrat-Charvillon and François-Xavier Standaert. Mutual information analysis: How, when and why? In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2009.

[WO11] Carolyn Whitnall and Elisabeth Oswald. A comprehensive evaluation of mutual information analysis using a fair evaluation framework. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 316–334, 2011.

[WW04] Jason Waddle and David Wagner. Towards efficient second-order power analysis. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2004.

Publication

DPA, masking and bitslicing at 1 GHz

Publication Data

Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1 GHz. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 599–619. Springer, 2015.

Our contribution

One of the three main authors.

DPA, Bitslicing and Masking at 1 GHz

Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede

KU Leuven Dept. Electrical Engineering-ESAT/COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
{firstname.lastname}@esat.kuleuven.be

Abstract. We present DPA attacks on an ARM Cortex-A8 processor running at 1 GHz. This high-end processor is typically found in portable devices such as phones and tablets. In our case, the processor sits in a single board computer and runs a full-fledged Linux operating system. The targeted AES implementation is bitsliced and runs in constant time and constant flow. We show that, despite the complex hardware and software, high clock frequencies and practical measurement issues, the implementation can be broken with DPA starting from a few thousand measurements of the electromagnetic emanation of a decoupling capacitor near the processor. To harden the bitsliced implementation against DPA attacks, we mask it using principles of hardware gate-level masking. We evaluate the security of our masked implementation against first-order and second-order attacks. Our experiments show that successful attacks require roughly two orders of magnitude more measurements.

Keywords: Side-channel analysis, DPA, ARM Cortex-A8, bitslicing, gate-level masking.

1 Introduction

Side-channel attacks allow to extract secrets, such as cryptographic keys or passwords, from embedded devices with relatively low effort. Kocher reported in his seminal paper [Koc96] extracting cryptographic keys from the observation of the execution time taken by an implementation of Diffie-Hellman, RSA or DSS. A common characteristic of side-channel attacks is that they target concrete implementations, and thus they are oblivious to the intrinsic mathematical security of the algorithm. They can be readily applied to implementations of algorithms that are resistant to traditional mathematical attacks.

Apart from timing, many other side-channels have been discussed in the literature. Most notably, the instantaneous power consumption is a powerful side-channel for embedded devices [KJJ99], and efficient exploitation mechanisms, such as Differential Power Analysis (DPA), are known. DPA requires access to the target

device to collect a number of instantaneous power consumption traces while the device is running the cryptographic implementation. The key can be derived from the statistical analysis of the power consumption traces.

A popular variant of power analysis attacks are Electromagnetic Analysis (EMA) attacks [GMO01, QS01]. EMA attacks measure the electromagnetic emanations from the device and subsequently apply similar statistical techniques as DPA. An advantage is that electromagnetic measurements do not require to establish electrical contact, thus EMA can be less invasive than conventional power analysis.

Side-channel attacks on small embedded devices, such as microcontrollers and cryptographic co-processors, are nowadays a well-understood threat and a fruitful field of academic research. However, there are only a few studies of side-channel attacks on more powerful general-purpose systems. This is highly relevant to the gradual paradigm shift towards moving the cryptographic operation to the main processor, as proposed in mobile payments and host card emulation.

In this paper we investigate the DPA susceptibility of block-cipher implementations on high-end embedded devices. As an illustrative test case, we focus on the Advanced Encryption Standard [AES01] (AES) and an ARM Cortex-A8 processor. This processor core is found in portable consumer electronic devices, such as phones (Apple iPhone4, Samsung Galaxy S, Google Nexus S), tablets, set-top boxes, multimedia entertainment systems (Apple TV, Apple iPod Touch 4th gen), home networking or storage appliances and printers.

The Cortex-A8 is a powerful and complex processor that features significant differences with typical targets of side-channel attacks. It is a 32-bit processor with a 13-stage pipeline, dynamic branch prediction, L1 and L2 cache memories, a rich ARMv7 instruction set and a separate SIMD execution pipeline and register file (NEON). It can run at up to 1 GHz clock frequency. At the software level, there is normally a full multi-tasking operating system with shared resources, different competing processes and interrupts. It is not clear if DPA can be successfully applied to such target devices. One goal of our work is to fill this gap.

1.1 Related work

AES on high-end embedded devices. An efficient option for AES software implementations on high-end processors is the *T-table* approach due to Daemen and Rijmen [DR02]. Its core idea is to merge three of the four AES transformations (`SubBytes`, `ShiftRows` and `MixColumns`) into four lookup tables. At the cost of storing 4 kbytes, this method allows to compute an AES-128 encryption using only 160 table lookups and 44 XOR operations. Since the four lookup tables are rotations of each other, it is possible to reduce the memory requirements to 1 kbyte

by storing a single table. For architectures with inline barrel shifter such as ARM, this characteristic can be used without performance loss [OBSC10].

While efficient, implementations based on lookup tables are a target for side-channel attacks on processors with cache memories. Exploiting cache-related timing variabilities was already mentioned by Kocher [Koc96], and further elaborated on by Kelsey *et al.* [KSWH00] and Page [Pag02]. In recent years, several practical attacks against the T-table AES implementation of OpenSSL have been published, see for instance the works of Bernstein [Ber05], Bonneau and Mironov [BM06] and Osvik *et al.* [OST06]. The root of the problem stems from the difficulty to load array entries into the CPU registers without this depending on the index pointer. As suggested by Bernstein *et al.* [BLS12], a secure library should systematically avoid loads from addresses that depend on secret data. While one could always resort to *computing* the AES S-Box to achieve constant execution time, the performance penalties of straightforward implementations would be considerable.

It is in this context that bitsliced implementations rise as an attractive alternative for AES in software. Originally proposed by Biham [Bih97] to improve the performance of DES in software, the idea behind bitslicing consists in describing a cryptographic algorithm as a sequence of Boolean operations which can be implemented with only bitwise instructions. Since there are no table lookups, bitsliced implementations are inherently resilient to cache timing attacks. The first bitsliced software implementation of the AES for x64 processors is due to Matsui [Mat06]. An alternative implementation for 64-bit platforms is presented by Könighofer [Kön08]. The advent of Single Instruction Multiple Data (SIMD) extensions on Intel Core2 processors has enabled a more efficient usage of the 128-bit XMM registers. Matsui and Nakajima [MN07] were first to take advantage of this and proposed a high-speed bitsliced implementation of the AES at 9.2 cycles/byte, albeit conditioned to input data blocks of 2 kbytes. More recently, Käsper and Schwabe [KS09] proposed the fastest implementations of AES-CTR and AES-GCM up to date, running at 7.59 cycles/byte and 10.68 cycles/byte, respectively.

Side-channel attacks on high-end embedded devices. With the notable exception of cache timing attacks, the susceptibility of high-end embedded processors to side-channel attacks has received only little attention in the literature, particularly when compared to the attention that has been given to less complex platforms. Gebotys *et al.* [GHT05] showed how to attack Java implementations of AES and ECC running on a PDA equipped with a "mid-range" 32-bit ARM7TDMI at 40 MHz. The authors performed a differential EMA attack in the frequency domain in order to deal with the issue of trace misalignment. A follow-up work by Aboulkassimi *et al.* [AAF+11] similarly used differential EMA to attack AES implementations. The target device was a mobile phone with a 32-bit processor running at 370 MHz. Kenworthy and Rohatgi [KR12] applied Simple Power Analysis (SPA) and leakage detection techniques to show the susceptibility of several implementations to EMA. Although no processor frequency is specified, the

acquisition bandwidth of the setup used was limited to 60 MHz. Finally, Nakano *et al.* [NSN+] performed SPA attacks on ECC and RSA implementations running on an Android Smartphone clocked at 832 MHz.

Masking countermeasures. A popular and well-studied countermeasure to thwart power analysis attacks is masking [CJRR99, GP99]. Contrary to other approaches, masking is a provable sound countermeasure and widely employed in practice. In its simplest form, masking consists of splitting every key-dependent intermediate s that appears throughout the computation into two shares (s_1, s_2) such that $s_1 \star s_2 = s$. The group operation $\star$ is typically XOR. The splitting is such that each share s_i is statistically independent of the intermediate s. This condition should be preserved throughout the entire masked computation, and implies that knowledge of any individual s_i does not reveal any information about the intermediate s, and thus about the key.

Masking can be applied at different abstraction levels: from the algorithmic level (public key cryptography algorithms [Cor99, MDS99] as well as symmetric key algorithms such as DES [GP99] or AES [AG01]) to the gate level [Tho01]. Algorithm-level masking can result in more compact implementations. However, this masking method is not a general approach as it is tied to a specific algorithm. On the other hand, gate-level masking performs the splitting at the bit level and provides the implementer with a set of secure logic gates to compute on. It it thus a versatile method to securely implement any given circuit.

1.2 Contributions

Our first contribution is to investigate the feasibility of DPA attacks on modern gigahertz embedded processors. Our experimental platform is a Sitara ARM Cortex-A8 32-bit RISC processor mounted on a Beaglebone Black (BBB) platform and running a complete Ångström Linux distribution. Our test application is a bitsliced implementation of AES-128 encryption immune to cache timing attacks. Our experiments show that the most difficult part of an attack is of practical nature (measurement point, triggering, alignment) and that basic DPA attacks succeed with a few thousand measurements. For the sake of reproducibility, we describe all steps carried out in our analysis in detail.

Our second contribution is to apply gate-level hardware masking to protect our implementation. We show that it is not difficult to equip an unprotected bitsliced implementation with masking. In addition we fully implement a masked AES on the same platform and test its resistance to first-order and second-order attacks. Our experiments show that breaking our masked implementation requires roughly two orders of magnitude more measurements than breaking the unprotected implementation.

2 A bitsliced AES implementation

Our test application is a bitsliced implementation of the AES based on the construction of Könighofer [Kön08]. We adapted it for our 32-bit processor. Note that this is a poor decision if one aims for performance, i.e. bitsliced implementations pay off in software contexts only if the target processor contains large (and possibly many) registers. Nevertheless, our aim is neither to propose nor to achieve high-throughput implementations, but rather analyze bitsliced implementations from the DPA-security standpoint. In fact, and as will become clear later, our insights also apply for larger wordsize architectures such as e.g. NEON.

Hardware description of AES. In the following we focus on AES-128 encryption. The first step towards a bitsliced description consists in describing all cipher transformations (`AddRoundKey`, `SubBytes`, `ShiftRows` and `MixColumns`) as a fixed sequence of Boolean operations. The goal is to employ only bitwise operations i.e. an equivalent gate representation in hardware contexts. The main difficulty of this process consists in finding an efficient way to compute the non-linear part of the AES S-Box.

There exist many hardware flavours of AES depending on whether they aim for throughput, area, low-power, etc. For bitsliced contexts, we are interested in *compact* implementations. Most successful designs in this direction compute the inverse in $GF(2^8)$ using subfield arithmetic, as originally suggested by Rijmen [Rij01]. This is the case of the works due to Rudra *et al.* [RDJ+], Wolkerstorfer *et al.* [WOL02] and Satoh *et al.* [SMTM01], the latter building also on the tower-field representation of Paar [Paa94]. As in [Kön08], we employ the AES S-Box representation by Canright [Can05] illustrated in Figure 1.

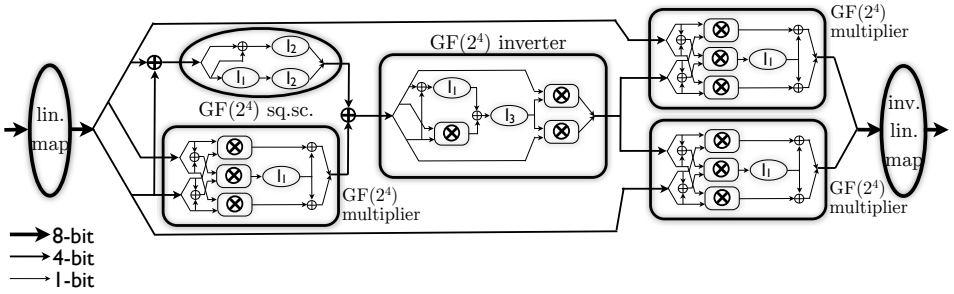


Figure 1 – AES S-Box representation due to Canright.

Bitsliced format. In the standard AES representation a 128-bit input message block A is described as a 4×4 byte matrix. This is illustrated in the upper left hand side of Figure 2. Each byte is addressed as A_i. The cipher transformations

are commonly defined at byte level in order to operate on the matrix representation of the state, e.g. either at element level (`AddRoundKey` and `SubBytes`), at row level (`ShiftRows`) or at column level (`MixColumns`). This representation is however inadequate for bitsliced implementations, as all steps are defined at bit level. Therefore, one needs to find a different representation of the cipher state. The most straightforward option consists in arranging the state as a vector of 128 elements, each corresponding to a bit. This choice is however unsuitable in practice, as the state cannot be fully kept in registers and memory accesses easily become a major bottleneck.

An alternative bitsliced representation uses a more compact state of 8 elements [KS09, Kön08, Mat06], each containing a particular bit of the 16 state bytes A_i. Let us denote the bitsliced state elements by $\mathcal{R}_i$. Going to the bitsliced domain requires to split the bytes A_i and store the bits, from LSB to MSB, to the corresponding registers, $\mathcal{R}_1$ to $\mathcal{R}_8$. Note that one input message block A fills only 16 bits in each $\mathcal{R}_i$. Therefore several input messages can be processed in parallel, e.g. by storing bits from several input message blocks into each register. This is illustrated in Figure 2 for the case of 32-bit registers. In this case, a second plaintext B is processed concurrently with A.

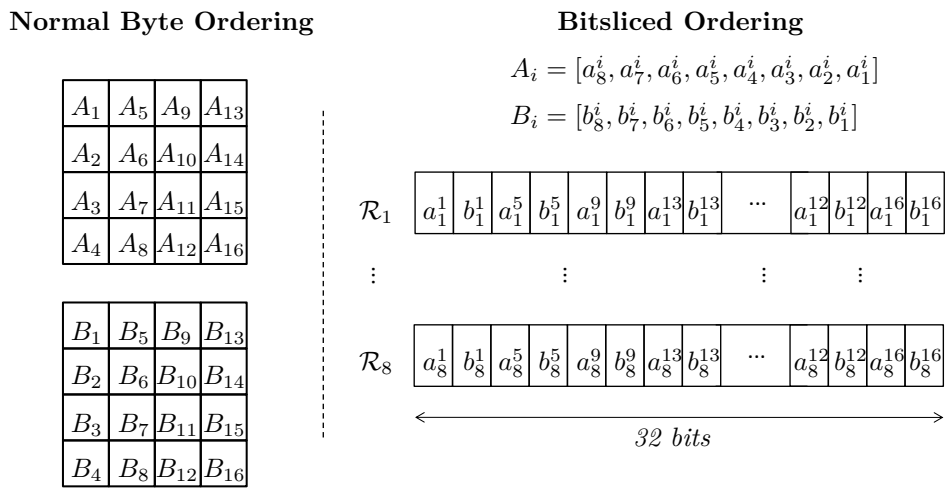


Figure 2 – Layout of bitsliced AES registers.

Coding style. We have coded our bitsliced AES implementation in C language and mimicked the concept of hardware gates by using software macros for all atomic operations. This approach allows us to write our program as a fixed sequence of calls to five main macros: bitwise operations (`XOR`, `AND` and `NOT`), data transfers (`MOV`) and left rotates (`ROTL`):

```
#define XOR(c,a,b)      c = a ^ b;
#define AND(c,a,b)      c = a & b;
#define NOT(c,a)        c = ~ a;
#define MOV(c,a)        c = a;
#define ROTL(c,a,l)     c = (a ≪ l) | (a ≫ (32 - l));
```

The main benefit of this approach is that protecting the implementation with gate-level masking requires only rewriting the macros. This point will be elaborated on in Section 4.

3 Developing an attack

The BBB is a complex single board computer. The main component is a high-performance TI AM3358 Sitara System on Chip (SoC) based on the ARM Cortex-A8 core. To give an idea of the complexity, we point out that the main processor can be clocked up to 1 GHz and that the SoC features a DDR3 memory controller, a 3D graphics engine, a vast array of peripheral support (incl. USB, ethernet) and two 32-bit sub-processors (technically, programmable real-time units) for time-critical tasks, among others.

3.1 Strategies for side-channel measurements

From all the components of the single board computer, we are mostly interested in the side-channel leakage of the ARM processor. An obvious way to access it would be to measure the SoC's power consumption. However, performing a power measurement on a BGA package is not straightforward, as the pins are covered by the package and not easily accessible.

A second strategy might be to measure the power consumption of the entire single board computer. At least, doing the actual measurement should not be difficult as the entire board is powered by a single 5 V supply (via USB or a dedicated connector, e.g. if more than 500 mA is needed). But we expect the global power consumption to be very noisy (many active components on the board). Furthermore, there is a dedicated power management IC and numerous decoupling capacitors between the power supply and the SoC. The high operating frequencies of the processor require capacitor banks that can deal with low, medium and high frequencies.

The third approach is the one we actually followed. We opted for a "contact-less power measurement". To clarify, others have used the same technique and called it "electromagnetic measurement". We do use an electromagnetic pen probe but we do not aim at measuring emanations from the ARM processor. Rather, we measure the EM field around different components on the board that are somehow involved in the current loop to the ARM core. In general, voltage regulators and decoupling

capacitors [DO09, OC12] are promising candidates. In our case, the dedicated power management IC is quite complex and physically located far away from the ARM core. For these reasons, we do not think that it would provide a useful signal. Therefore the decoupling capacitors are the best candidates. In general, the closer the capacitor is to the processor, the better signal it can provide. In summary, we use an electromagnetic probe to measure a signal that is correlated with the chip's power consumption. We therefore think of the technique as a contact-less power measurement.

3.2 Experimental setup

Our experimental setup comprises:

- a stock BBB platform running a complete Linux Ångström distribution. We did not modify the software or the hardware and operate the board in its factory configuration. The Linux distribution is based on Debian 7 with kernel version `3.8.13-bone47 (root@imx6q-wandboard-2gb-0)`. This is a preemptive multitasking operating system with plenty of simultaneously running processes. We did not switch off any running service. The command `ps aux` reports 102 processes running on the system. Among others, we found running the `Xorg` graphical server (with the onboard HDMI driver output activated), the `apache2` webserver (including the `nodejs` server-side javascript runtime environment, to our surprise) and the `sshd` server (with an open session running throughout all experiments for monitoring purposes). We power the board via the USB connection from the measurement PC. We did not make any effort to supply the board with a particularly clean voltage. The board is connected to the measurement PC via ethernet-over-USB. We did not disable the blinking blue leds that indicate activity.
- A Langer magnetic near field probe, model RF-B. The reported frequency bandwidth is from 30 MHz to 3 GHz [EMV].
- A wideband 30 dB low-noise amplifier from Langer, model PA 303. The reported frequency bandwidth goes from DC to 3 GHz.
- A Tektronix DPO70404C 8-bit oscilloscope with an analog bandwidth of 4 GHz. Most of the time we sampled at 6 GS/s to make full use of our setup's bandwidth (Nyquist rate).

We use the Linux command `cpufreq-set -f 1000 MHz` to bring the system into a high-performance state. In this state the board cannot enter low-power mode and the processor core is permanently clocked at 1 GHz, which is well within the bandwidth of our measurement setup. Figure 3 shows a photo of our experimental setup (left) and a representation of our EM probe and the field orientation it is able to register (right) [EMV].

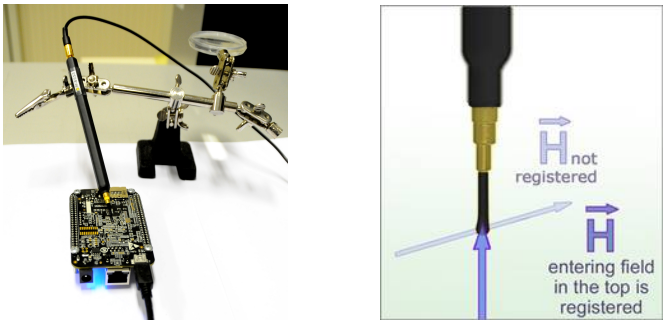


Figure 3 – Photo of our setup (left) and EM antenna schematic (right).

3.3 Approach

Our first step is to find a suitable measurement position for the EM antenna. The EM antenna tip is small enough (we measured a diameter of 2 mm) to allow us to get in between components and to measure individual components' EM field without picking up too many signals from neighboring components. For the purpose of locating an appropiate position for the EM antenna, we wrote a short C program that exercises memory accesses and ran it on the BBB in a loop. The program executes 1000 NOPs, then repeatedly fills a buffer in memory with the value `0x00000000` and then with the value `0xffffffff` 1000 times, followed by again 1000 NOPs. We manually move the antenna over the PCB surface and slowly from component to component. We carefully monitor the sampled signal on the oscilloscope for a pattern that looks correlated to the execution of our C code. We begin doing this by trial and error, and we focus on the capacitors in the SoC's power supply as explained above. Note that this is a tedious task because we need to get not only the probe's tip in the right *location* but we also need to get the probe in the right *orientation* (see Fig. 3 right). As the search was very time consuming we had a look at the BBB PCB schematics [bbb]. We identified a bank of capacitors in the SoC's VDD core supply. They should be good candidates for measurement points as their EM fields should contain a lot of useful signal about the processor core's activity. Next we locate these decoupling capacitors on the PCB and manually scan them with the EM probe one by one.

We did not find a useful signal around these capacitors (that does not mean there is no useful signal) and reverted to trial and error testing of other decoupling capacitors in the SoC's supply network. Eventually we found a good signal near C66 (see Appendix A in the full version of this paper [BGRV15]), a 0.1 μF multilayer ceramic capacitor in a 3.3 V supply rail. We used this probe position and orientation for all measurements and did not further explore the board for other useful signals.

Now that we found a suitable measurement point, the next step is to deal with the timing and triggering. We run our bitsliced implementation of AES-128 encryption from Section 2 on the BBB. We can send the inputs from the measurement PC and read back the outputs as well. Recall that we keep an SSH connection open between the measurement PC and the BBB for this purpose and for monitoring. After some trial and error work we find a good pattern (related to I/O) to trigger the oscilloscope on. Figure 4 shows the plot of an overview measurement. Note that we need to substantially lower the sampling rate for some of these long measurements. The execution of our C program causes the dense pattern in the middle of the plot. The isolated peaks left and right of the dense pattern are caused by other processes.

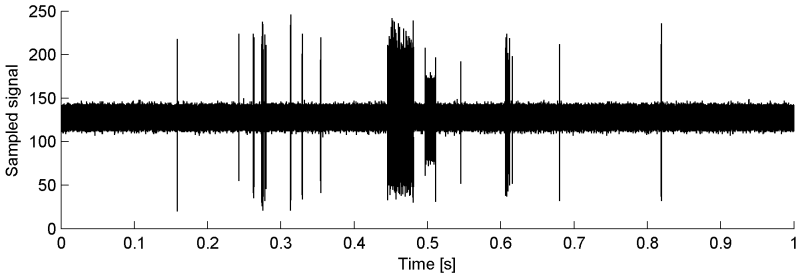


Figure 4 – Overview measurement of our unprotected AES.

Figure 5 shows the plot of a more focused measurement. We see patterns caused by the reception of 34 bytes (two plaintext blocks of 16 bytes each and two control words) followed by patterns caused by the AES encryption in the middle of the figure. We do not know what causes the "block" pattern on the figure's right hand side.

Figure 6 shows a zoom on the patterns caused by the AES operation. It is tempting to let the human eye search for patterns of the ten AES rounds, but in fact the AES makes only a small part of this measurement (as marked by the dotted red rectangle in the figure). We are not sure what the other processing is. We know that some of it is the conversion of plaintexts to the bitsliced format, and from bitsliced format to ciphertexts.

Figure 7 shows a plot of a measurement with the actual AES-128 encryption in the middle of the plot. One would expect to see a sequence of nine very similar patterns (the first nine AES rounds) followed by a different pattern (the tenth round without `MixColumns`). However, in this figure we see a sequence of only eight very similar patterns followed by a different pattern. It seems that our measurement is missing one normal round. Our experiments confirm that what we recognize corresponds to rounds two to ten. The execution of the first AES round leads to a pattern that is more scattered over time, but it fills up the instruction cache so that the next rounds are executed much faster which leads to a more dense and clear pattern.

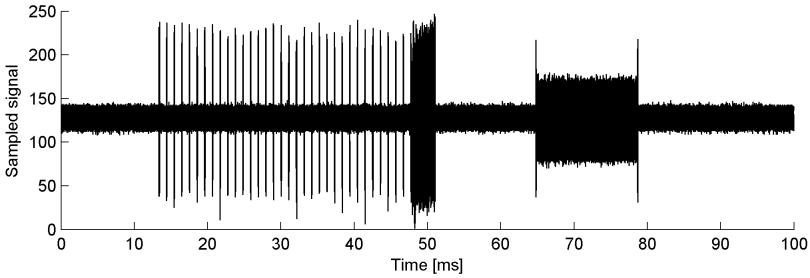


Figure 5 – Overview measurement of our unprotected AES.

When we execute several encryptions in a batch, the second and all following encryptions typically run from cache and show clear patterns also for the first round that we can use for alignment. Figure 8 shows the single-sided amplitude spectrum of a measurement. The spectrum shows a clear and sharp peak at 1 GHz. Even though we found a seemingly stable trigger, the measurements of the AES encryption are actually heavily desynchronized. Recall that we are working with a high-end ARM processor on a complex SoC and that our C program is only one of more than 100 running processes (and we do not run it with elevated priority!). Therefore, filtering out mis-triggered measurements and carefully aligning the remaining measurements is crucial. In fact, we spend about seven times more time on the post-processing than on the measurement.

3.4 Attack

We aim to break our unprotected implementation with a first-order correlation DPA attack [BCO04] against the first round. The next step is to try to find a pattern in the traces that is related to the (S-box computation in the) first round, and to align all useful traces on that pattern. Finally we try to attack the implementation with 10 000 aligned measurements.

We need to think about a power model because the implementation is bitsliced. A typical byte-oriented implementation uses an S-box table in memory and processes the AES state byte by byte. The key point here is that all 8 bits of an S-box output are computed (or looked-up) at the same time. Hence one can expect all bits of the S-box output to leak at the same time and this gives rise to the commonly used "Hamming weight of the S-box output" power model. This is different for our bitsliced implementation. The eight S-box output bits are computed one after the other and stored in eight different registers. So if we assume for a moment that the implementation processes one plaintext block at a time, each of the eight registers holds 16 bits. For instance register $\mathcal{R}_1$ stores the 16 LSBs of the 16 state bytes. With the usual divide and conquer approach we aim to recover the key

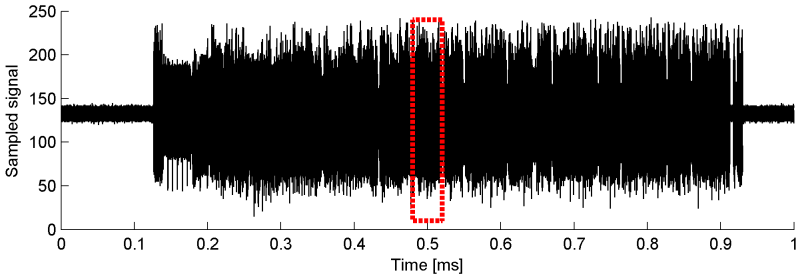


Figure 6 – Measurement of our unprotected AES.

byte by byte. If we make a guess about one key byte we can predict one S-box output but the eight bits are spread over eightregisters that are not processed at the same time. For a normal univariate attack we can therefore exploit only one bit effectively. The other 15 bits in the same register are algorithmic noise. If we want to exploit more bits in the same register we need to guess more key bytes, which quickly becomes computationally expensive. Alternatively we can think to attack each of the eight bits of one S-box output separately and then perform some majority voting, but we did not investigate this approach.

Now in our implementation the situation is similar but it actually processes two plaintext blocks in parallel. This means for instance that register $\mathcal{R}_1$ stores 32 LSBs, 16 of one plaintext and 16 of the other. As the key is fixed both plaintext blocks get encrypted under the same key. Making a guess on one key byte we can attack both encryptions at the same time and predict two bits in a register (2 out of 32 instead of 1 out of 16 in the example above). Our power model is hence the Hamming weight (HW) of two bits in a register that are affected by the same sub-key. We stress that this observation has an *important consequence*: processing more plaintext blocks in parallel does not make an attack harder if the adversary is aware of the bitsliced implementation. In fact, the ratio of predicted bits and processed bits, and hence the ratio of signal to algorithmic noise, is constant.

Figure 9 shows an exemplary result of a 2-bit attack against one key byte. The plot on the left hand side shows the correlation traces for all key hypotheses obtained using 10 000 measurements. The trace for the correct key hypothesis is plotted in black. The plot shows that the correct key hypothesis leads to a distinguishable and clear correlation peak. The plot on the right hand side shows the highest and lowest correlation value for each key hypothesis (from the overall time frame) over the number of measurements. In addition we also plot the 99.99% confidence interval for sample correlation equal to zero (dashed lines). The plot shows that only few thousand measurements are required for the correct key (black line) to stand out and hence for the attack to succeed.

Attacks targeting the same (other) key byte(s) using the leakage of other (the

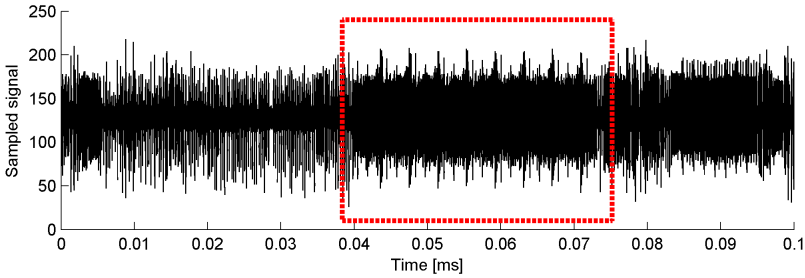


Figure 7 – One of the first measurements of our unprotected AES.

same) register(s) give very similar results. Surprisingly full key recovery is hence possible using only a few thousand measurements!

4 Masking a bitsliced AES implementation

Since a bitsliced software implementation mimics a hardware circuit, gate-level masking appears as a very attractive candidate to protect our AES implementation. Applying gate-level masking to an already existing implementation can actually be done in a pretty straightforward manner. It only requires to protect the design's elementary Boolean functionalities, while the original sequence of operations remains unmodified. A direct consequence of this is that any optimization performed in the unprotected implementation, e.g. to improve the design's throughput, is automatically inherited by the protected implementation.

Generally, linear functions such as the XOR gate, are trivial to mask by just computing on each share independently. The challenging part of gate-level masking is to provide a construction for non-linear gates. One of the first works tackling this problem is due to Trichina [Tri03]. Trichina gives a secure AND gate that takes two shares of each input bit a, b and produces two output shares of $c = a \cdot b$. The secure AND gate consumes one fresh random bit r. If the input bit a (resp. b) is shared into a_1 and a_2 (resp. b_1 and b_2), the two output shares c_1, c_2 of the Trichina gate are defined as

$$c_1 = r \tag{1}$$

$$c_2 = (((a_1 b_1 \oplus r) \oplus a_1 b_2) \oplus a_2 b_1) \oplus a_2 b_2. \tag{2}$$

It is easy to verify that this AND gate description is correct, namely, that the output shares XOR to $a \cdot b$. The description is also secure against first-order attacks: each variable occurring during the execution is independent of any unshared value

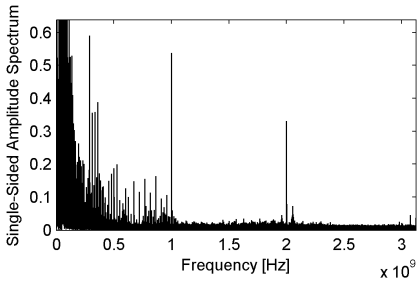


Figure 8 – Single-sided amplitude spectrum of a measurement.

a, b or c. Note however that the order of partial computations of c_2 is relevant for the security of the gate.

Masked bitsliced format. Applying first-order Boolean masking requires to split any sensitive intermediate variable s into two shares such that $s = s_1 \oplus s_2$. For our implementation, this implies that each of the eight original state registers $\mathcal{R}_i$ becomes a pair $(\mathcal{R}_i^1, \mathcal{R}_i^2)$ such that $\mathcal{R}_i = \mathcal{R}_i^1 \oplus \mathcal{R}_i^2$. We denote $\mathcal{R}_i^1$ as mask state and $\mathcal{R}_i^2$ as masked state. The plaintext in bitsliced format is shared in this way at the beginning of the execution.

Masked operations. Our original bitsliced implementation employs only five operations which are described as macros. These are: `XOR`, `AND`, `NOT`, `MOV` and `ROTL`. Our masked implementation substitutes each occurrence of these by its secure equivalent: `SXOR`, `SAND`, `SNOT`, `SMOV` and `SROTL`, respectively. The secure equivalents operate sequentially on the mask state $\mathcal{R}_i^1$ and the masked state $\mathcal{R}_i^2$. The implementations of `SXOR`, `SMOV` and `SROTL` are trivial, as they consist of implementing the corresponding `XOR`, `MOV` or `ROTL` twice: one for $\mathcal{R}_i^1$ and another for $\mathcal{R}_i^2$. In a similar way, the secure `SNOT` is simply computed by applying `NOT` to one of the shares. The implementation of `SAND` is more elaborate and follows closely the lines of the Trichina gate. A circuit representation of the gate is shown in the left part of Figure 10, while its macro representation is given in the right part of Figure 10.

In contrast to the original bitsliced macros, each variable in the macro is now an array of 2 elements: mask state and masked state. Each of the two input arrays (`a`, `b`) is thus composed of two registers (`a[0]`,`a[1]`) and (`b[0]`,`b[1]`), respectively. Two temporal registers (`t0` and `t1`) are additionally used to preserve the correctness of the macro in case one of the source registers is also the destination, e.g. to prevent errors when the macro is called as `SAND(a,a,b)`. The result is placed in the output array `c` composed of registers `c[0]` and `c[1]`.

Randomness generation. The two operations that require randomness in the

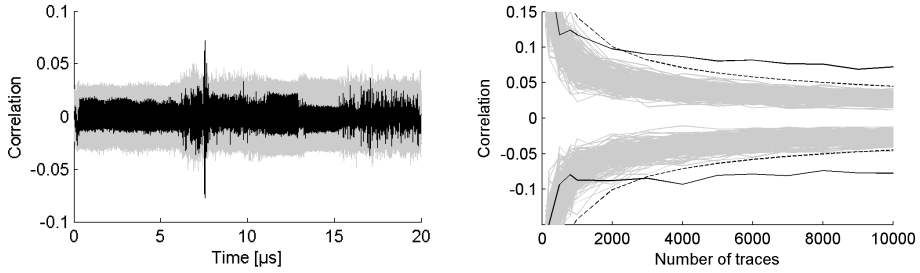


Figure 9 – Result of attack against unprotected implementation.

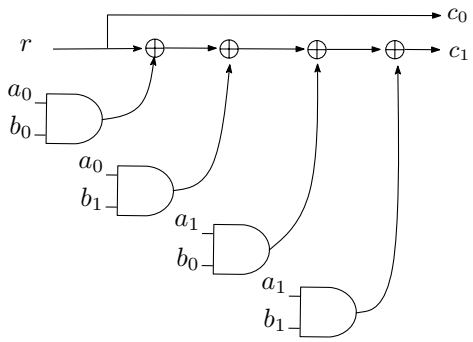
masked bitsliced implementation are the initial plaintext sharing and each `SAND` operation. We use the kernel's `/dev/urandom` cryptographic RNG to obtain the required randomness. We do not read a single byte each time a random byte is needed, instead, we read a chunk of randomness and place it in an internal buffer at the beginning of each encryption. Then, during the actual encryption the randomness is simply taken from this internal buffer. We implemented this mechanism to minimally interrupt the execution of the encryption and get clean measurements.

We note that masking typically does not need cryptographically strong random numbers for the masks. Although we used a cryptographically strong source of randomness for the masks, a lighter RNG can be used if needed, e.g. for performance reasons. When we later report that the RNG is switched off, we fill the internal buffer from `/dev/zero` instead of from `/dev/urandom`.

Performance. We have compiled our implementations directly in the BBB using the compiler version available in the Ångström Linux distribution, i.e. gcc version 4.6.3. No special flags have been used. The throughput loss of the protected implementation is roughly a factor 5 compared to the unprotected implementation. Further, the RAM usage increases by 32 bytes because of doubling the register state size. Our internal buffer for storing random numbers holds 2048 bytes. The only macro in our implementation that consumes randomness is `SAND`, which is used 37 times during the calculation of `SubBytes`. Taking into account that 32 bytes are required to mask the input plaintexts, this gives $32 + 10 \times (37 \times 4) = 1512$ random bytes per AES execution, or equivalently, 756 bytes per plaintext block.

5 Evaluation of masked implementation

In this section we evaluate the DPA resistance of our bitsliced and first-order masked AES implementation.




```
#define SAND(c, a, b){
  t0 = a[0] & b[0];
  t1 = a[0] & b[1];
  c[0] = RAND();
  t0 = t0 ^ c[0];
  t0 = t0 ^ t1;
  t1 = a[1] & b[0];
  t0 = t0 ^ t1;
  t1 = a[1] & b[1];
  t0 = t0 ^ t1;
  c[1] = t0;}
```

Figure 10 – Left: Trichina construction for the masked AND gate. Right: pseudocode for the `SAND` operation following the Trichina AND construction.

5.1 Attack when RNG is off

We first attack the implementation with the RNG switched off. In this case the implementation is effectively unprotected and we aim to break it with the same first-order attack as before: we guess one key byte and use the HW of the two affected bits in a register as power model. Since the code is different, the shape of the measurements is different as well, and we need to work through trial and error again to find a good pattern for trace alignment. And this is where the fact that the implementation is effectively unprotected helps. We should be able to break it easily, and we can use the attack result to judge and improve the discarding of mis-triggered measurements and the alignment step until we are satisfied. As a side note we also mention that we need to take longer measurements because in particular the S-box computation takes more time.

Figure 11 shows the result of an exemplary attack against one key byte. The plot on the left hand side shows that using 10 000 measurements the correct key hypothesis leads to a clear correlation peak. The plot on the right hand side confirms that, if the RNG is switched off, our masked implementation is as insecure as the unprotected implementation and can be broken with a few thousand measurements. Also in this case attacks targeting other key bytes and using the leakage of other registers give similar results. Full key extraction with a first-order attack is possible with a few thousand measurements.

5.2 Attack when RNG is on

Now we switch on the RNG and evaluate how much protection our masked implementation provides. Having performed the attacks with the RNG switched

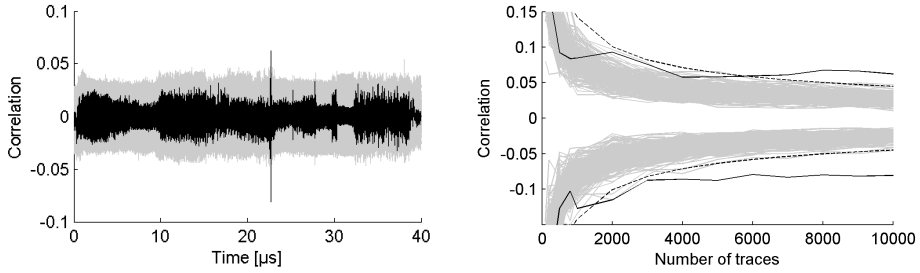


Figure 11 – Result of attack against masked implementation with RNG off.

off has two important advantages. First, we can keep all settings for triggering, for discarding mis-triggered measurements, and for alignment because the executed code is exactly the same and the general shape of the measurements does not change. And second, we know exactly when the S-box computations are performed and we can therefore narrow down the time window to analyze (including some margin).

It is well known that implementing masking securely in software is very difficult, and we do not expect our first attempt to mask a bitsliced implementation to provide a high level of resistance to attacks. Nevertheless, to ensure that we have enough measurements at hand to break our implementation we acquired 2 000 000 measurements. We stress that trace acquisition is rather quick, but in contrast to most academic works we have to deal with the computationally intensive and hence slow post-processing (discarding mis-triggered measurements and alignment). After post-processing we are left with about 1.2 million aligned measurements.

We applied the same 2-bit first-order DPA attack as before in various settings, targeting different key bytes and registers. The results differ a lot depending on the specific setting. To give an idea of the range, we provide two results in Figure 12. They target different key bytes and registers. While in the upper plots the attack clearly succeeds and requires about 600 000 measurements, the attack in the lower plots fails even if using 1.2 million traces. Nevertheless, we confirmed that, using alternative combinations of target key byte and register, full key extraction with first-order DPA and using 1.2 million measurements is possible.

Considering the well known difficulties with masking in software and the surprisingly easy attacks against the unprotected implementation, we expected attacks against our masked implementation to succeed with much less traces. Our results are therefore promising and good news for the idea to combine bitsliced software and gate-level masking. Recall that our implementations are coded in C, processed by a compiler and we have little control over the code that will be eventually executed. Also, we stress that this is our first attempt to mask the implementation. The

fact that the result of each individual Boolean operation is registered in a bitsliced implementation probably helps. Glitches that are an issue for masked *hardware* implementations [MPG05] are no threat here.

We also performed a few exemplary univariate and bivariate second-order attacks using all available 1.2 million measurements. Concretely, we processed the measurements to combine each pair of time samples using the absolute difference combination function [Mes00] or the centered product combination function [PRB09]. This combination step yields a combinatorial blow-up in the number of time sample pairs to be analyzed jointly and makes these attacks very computationally expensive.

We then applied the same 2-bit attack to the combined measurements. Figure 13 depicts the results of an exemplary attack using the absolute difference combination function. The plot on the left hand side shows the maximum absolute value of the correlation coefficient for each key byte hypothesis across *all pairs* of time samples. The correct value (indicated by a dashed vertical line) clearly stands out against all competing hypotheses. For the plot on the right hand side we restrict the analysis to the single pair of time samples for which the correct key guess gives maximal correlation. It shows that the attack can be successful starting from around 400 000 measurements if the adversary already knows which pair of time samples to analyze. In other words, a more realistic attack will very likely require more measurements to succeed. However, a more extensive analysis over all pairs of time samples is computationally expensive.

Our results lead to two interesting observations. First, the second-order attack only works when we use the absolute difference combination function. A similar attack using the centered product combination function is unsuccessful. This is in contrast with theoretical results proving the optimality of the centered product combination function for second-order attacks [PRB09] assuming Hamming weight leakage and Gaussian noise. We speculate that our scenario does not meet these assumptions sufficiently and that the absolute difference combination function is more robust. And second, the number of measurements required for a successful first-order attack is not substantially lower than the number of measurements needed for our "idealized" second-order attack. A converse result would indicate a flaw in the masked implementation.

6 Conclusion

The threat of side-channel attacks to the security of microcontrollers and cryptographic co-processors appears to be well understood by both industry and academia. Yet the same cannot be said for high-end embedded processors as used in phones and tablets. In this situation one may naturally wonder whether such complex, high-performance devices operating in the GHz range and executing

multitasking operating systems are at all vulnerable to DPA. In this work we answer this question positively. By means of experiments we show that DPA attacks against constant-time bitsliced implementations of the AES running on a 1 GHz ARM Cortex-A8 processor are not only possible, but in fact rather easy to mount. The most challenging parts of an attack are triggering and trace alignment. Finally, we mask our implementation inspired by gate-level masking and evaluate its resistance against first-order and second-order DPA attacks. Our results indicate that the implementation is more secure than we anticipated and therefore highlight the potential of combining bitsliced software implementations and gate-level masking.

Acknowledgements. We would like to thank the CHES 2015 reviewers for their valuable feedback. This work has been supported in part by the Research Council of KU Leuven (GOA/11/007), by the Flemish Government FWO G.0550.12N and by the Hercules foundation (AKUL/11/19). Oscar Reparaz is funded by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO). Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO).

References

[AAF+11] Driss Aboulkassimi, Michel Agoyan, Laurent Freund, Jacques J A Fournier, Bruno Robisson, and Assia Tria. ElectroMagnetic analysis (EMA) of software AES on Java mobile phones. In *Information Forensics and Security - WIFS 2011*, pages 1–6. IEEE, 2011.

[AES01] Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards (FIPS) Publication 197, 2001.

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES, Secure against Some Attacks. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 309–318. Springer, 2001.

[bbb] BBB PCB schematics. http://elinux.org/Beagleboard:BeagleBoneBlack.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[Ber05] Daniel J Bernstein. Cache-timing attacks on AES, 2005.

[BGRV15] Josep Balasch, Benedikt Gierlichs, Oscar Reparaz, and Ingrid Verbauwhede. DPA, Bitslicing and Masking at 1˜GHz. Cryptology ePrint Archive, Report 2015/xyz, 2015.

[Bih97] Eli Biham. A Fast New DES Implementation in Software. In Eli Biham, editor, *Fast Software Encryption - FSE '97*, volume 1267 of *LNCS*, pages 260–272. Springer, 1997.

[BLS12] Daniel J Bernstein, Tanja Lange, and Peter Schwabe. The Security Impact of a New Cryptographic Library. In Alejandro Hevia and Gregory Neven, editors, *Progress in Cryptology - LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 159–176. Springer, 2012.

[BM06] Joseph Bonneau and Ilya Mironov. Cache-Collision Timing Attacks Against AES. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, volume 4249 of *LNCS*, pages 201–215. Springer, 2006.

[Can05] David Canright. A Very Compact S-Box for AES. In Josyula R Rao and Berk Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *LNCS*, pages 441–455. Springer, 2005.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 292–302. Springer, 1999.

[DO09] Abid Uveys Danis and Berna Ors. Differential power analysis attack considering decoupling capacitance effect. In *Circuit Theory and Design - ECCTD 2009*, pages 359–362, 2009.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[EMV] Langer EMV. Probe Specification. \urlhttp://www.langer-emv.com.

[GHT05] Catherine H Gebotys, Simon Ho, and C C Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In Josyula R Rao and Berk

Sunar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2005*, volume 3659 of *LNCS*. Springer, 2005.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic Analysis: Concrete Results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, 2001.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[Kön08] Robert Könighofer. A Fast and Cache-Timing Resistant Implementation of the AES. In Tal Malkin, editor, *Topics in Cryptology - CT-RSA 2008*, volume 4964 of *LNCS*, pages 187–202. Springer, 2008.

[KR12] Gary Kenworthy and Pankaj Rohatgi. Mobile Device Security: The case for side-channel resistance, 2012.

[KS09] Emilia Käsper and Peter Schwabe. Faster and Timing-Attack Resistant AES-GCM. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009*, volume 5747 of *LNCS*, pages 1–17. Springer, 2009.

[KSWH00] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side Channel Cryptanalysis of Product Ciphers. *Journal of Computer Security*, 8(2/3):141–158, 2000.

[Mat06] Mitsuru Matsui. How Far Can We Go on the x64 Processors? In Matthew J B Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *LNCS*, pages 341–358. Springer, 2006.

[MDS99] Thomas S Messerges, Ezzy A Dabbish, and Robert H Sloan. Power Analysis Attacks of Modular Exponentiation in Smartcards. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES'99*, volume 1717 of *LNCS*, pages 144–157. Springer, 1999.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç K Koç and C Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *LNCS*, pages 238–251. Springer, 2000.

[MN07] Mitsuru Matsui and Junko Nakajima. On the Power of Bitslice Implementation on Intel Core2 Processor. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *LNCS*, pages 121–134. Springer, 2007.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[NSN+] Yuto Nakano, Youssef Souissi, Robert Nguyen, Laurent Sauvage, Jean-Luc Danger, Sylvain Guilley, Shinsaku Kiyomoto, and Yutaka Miyake. A Pre-processing Composition for Secret Key Recovery on Android Smartphone. In *Information Security Theory and Practice - WISTP 2014*.

[OBSC10] Dag Arne Osvik, Joppe W Bos, Deian Stefan, and David Canright. Fast Software AES Encryption. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption - FSE 2010)*, volume 6147 of *LNCS*, pages 75–93. Springer, 2010.

[OC12] Colin O'Flynn and Zhizhang Chen. A Case Study of Side-Channel Analysis Using Decoupling Capacitor Power Measurement with the OpenADC. In Joaquín García-Alfaro, Frédéric Cuppens, Nora Cuppens-Boulahia, Ali Miri, and Nadia Tawbi, editors, *Foundations and Practice of Security - FPS 2012*, volume 7743 of *LNCS*, pages 341–356. Springer, 2012.

[OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. In David Pointcheval, editor,

Topics in Cryptology - CT-RSA 2006, volume 3860 of *LNCS*, pages 1–20. Springer, 2006.

[Paa94] Christof Paar. *Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields*. PhD thesis, University of Essen, 1994.

[Pag02] D Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. Cryptology ePrint Archive, Report 2002/169, 2002.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In Isabelle Attali and Thomas P Jensen, editors, *Research in Smart Cards - E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[RDJ+] Atri Rudra, Pradeep K Dubey, Charanjit S Jutla, Vijay Kumar, Josyula R Rao, and Pankaj Rohatgi. No Title.

[Rij01] Vincent Rijmen. Efficient implementation of the Rijndael S-box, 2001.

[SMTM01] Akashi Satoh, Sumio Morioka, Kohji Takano, and Seiji Munetoh. A Compact Rijndael Hardware Architecture with S-Box Optimization. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 239–254. Springer, 2001.

[Tho01] Larry Puhl Thomas S. Messerges Ezzat A. Dabbish. Method and apparatus for preventing information leakage attacks on a microelectronic assembly, 2001. US Patent 6,295,606.

[Tri03] Elena Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.

[WOL02] Johannes Wolkerstorfer, Elisabeth Oswald, and Mario Lamberger. An ASIC Implementation of the AES SBoxes. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002*, volume 2271 of *LNCS*, pages 67–78. Springer, 2002.

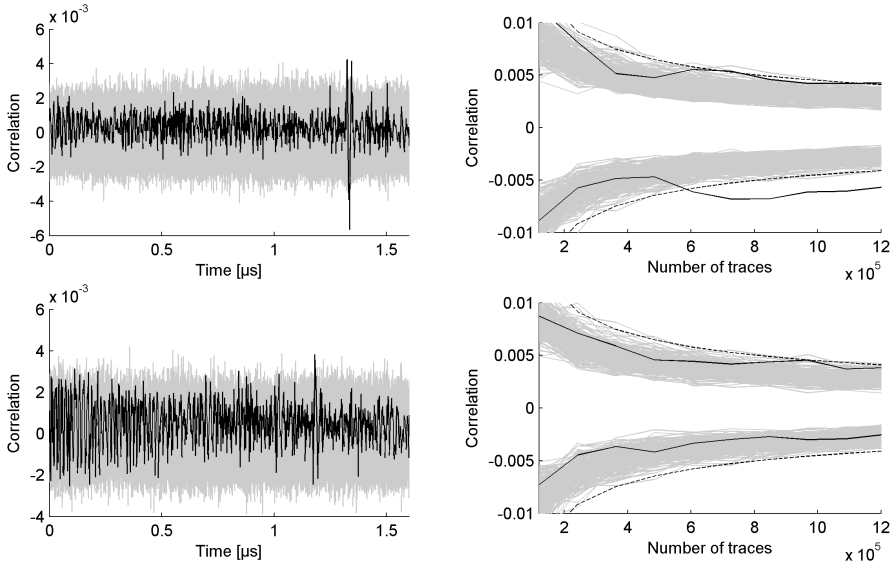


Figure 12 – Results of first-order attacks against masked implementation with RNG on.

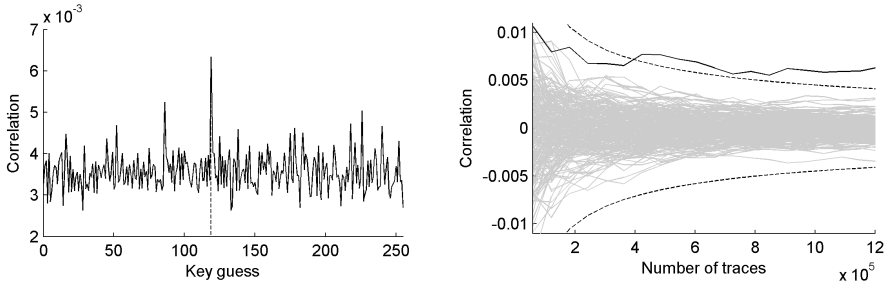


Figure 13 – Result of second-order attacks against masked implementation with RNG on.

Publication

A masked ring-LWE implementation

Publication Data

Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015.

Our contribution

Principal author, except for the FPGA implementation.

A masked ring-LWE implementation

Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede

KU Leuven Dept. Electrical Engineering-ESAT/COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Lattice-based cryptography has been proposed as a postquantum public-key cryptosystem. In this paper, we present a masked ring-LWE decryption implementation resistant to first-order side-channel attacks. Our solution has the peculiarity that the entire computation is performed in the masked domain. This is achieved thanks to a new, bespoke masked decoder implementation. The output of the ring-LWE decryption are Boolean shares suitable for derivation of a symmetric key. We have implemented a hardware architecture of the masked ring-LWE processor on a Virtex-II FPGA, and have performed side channel analysis to confirm the soundness of our approach. The area of the protected architecture is around 2000 LUTs, a 20% increase with respect to the unprotected architecture. The protected implementation takes 7478 cycles to compute, which is only a factor $\times 2.6$ larger than the unprotected implementation.

1 Introduction

Once the quantum computer is built, Shor's algorithm will make most current cryptographic algorithms obsolete. In particular, public-key cryptosystems that rely on number-theoretic hardness assumptions such as integer factorization (RSA) or discrete logarithms, either in $\mathbb{Z}_p^*$ (Diffie-Hellman) or in elliptic curves over finite fields, will be insecure. On the bright side, there is an entire branch of postquantum cryptography that is believed to resist mathematical attacks running on quantum computers.

There are three main branches of postquantum cryptosystems: based on codes, on multivariate quadratic equations or on lattices [BBD08]. Lattice-based cryptographic constructions, founded on the *learning with errors* (LWE) problem [Reg05] and its ring variant known as ring-LWE problem [LPR10], have become a versatile tool for designing asymmetric encryption schemes [LPR10], digital signatures [DDLL13] and homomorphic encryption schemes [FV12,BLLN13]. Several hardware and software implementations of such schemes have appeared in the literature. So far, the reported implementations have focused mainly on

efficient implementation strategies, and very little research work has appeared in the area of side channel security of the lattice-based schemes.

It comes as no surprise that implementations of postquantum algorithms are vulnerable to side-channel attacks. Side-channel attacks, as introduced by Kocher [Koc96], exploit timing, power consumption or the electromagnetic emanation from a device executing a cryptographic implementation to extract secrets, such as cryptographic keys. A particularly powerful side-channel technique is Differential Power Analysis (DPA), introduced by Kocher et. al. [KJJ99]. In a typical DPA attack, the adversary measures the instantaneous power consumption of a device, places hypotheses on subkeys and applies statistical tests to confirm or reject the hypotheses. DPA attacks can be surprisingly easy to mount even with low-end equipment, and hence it is important to protect against them.

There are plenty of countermeasures against DPA. Most notably, masking [CJRR99, GP99] is both a provably sound and popular in industry. Masking effectively randomizes the computation of the cryptographic algorithm by splitting each intermediate into several shares, in such a way that each share is independent from any secret. This property is preserved through the entire computation. Thus, observing any single intermediate (for example, by a side-channel, be it known or unknown) reveals nothing about the secret. However, there are not many masking schemes specifically designed for postquantum cryptography. In [BGL+14] Brenner et. al. present a masked FPGA implementation of the post-quantum pseudo-random function SPRING.

In the rest of the paper, we focus on protecting the ring-LWE decryption operation against side-channel attacks with masking. The decryption algorithm is considerably exposed to DPA attacks since it repeatedly uses long-term private keys. In contrast, the encryption or key-generation procedures use ephemeral secrets only [RRVV14].

Our contribution. In this paper we present a very compact masked implementation of the ring-LWE decryption function. The masking countermeasure adds very limited overhead compared to other previous approaches, thanks to a bespoke probabilistic masked decoder designed specifically for our implementation. We implemented the design on a Virtex-II FPGA and tested the side-channel security with practical experiments that demonstrate the validity of our approach.

Organization. The paper is structured as follows: we provide a brief mathematical background of the ring-LWE encryption scheme in Section 2 and describe a high-level overview of the proposed masked ring-LWE decryption in Section 3. In the next section we construct the masked decoder and in Section 5 we show the experimental results. We analyze the error rates of the decryption operation in Section 6 and apply error correcting codes. We dedicate Section 7 for the side channel evaluation.

2 Preliminaries

Notation. The Latin letters r, c_i indicate polynomials. When we want to explicitly access a coefficient of the polynomial we write $r[i]$. Multiplication of polynomials is written as $r * c_1$. Coefficient-wise multiplication is denoted as $r \cdot c_1$. The letter m denotes a string of bits, and q is an integer. Letters with prime x' or double prime x'' represent shares of variable x. Depending on the context, these shares are split either arithmetically $x = x' + x'' \pmod q$ or Boolean $x = x' + x'' \pmod 2$. A polynomial r is shared into (r', r'') by additively sharing each of its coefficients $r[i]$ such that $r = r' + r''$.

Ring-LWE. For completeness, we give in this section a description of the three major algorithms of the ring-LWE public-key cryptosystem [LPR10]: key-generation, encryption and decryption.

The ring-LWE encryption scheme works with polynomials in a ring $R_q = \mathbb{Z}_q[\mathbf{x}]/(f(x))$, where $f(x)$ is an irreducible polynomial of degree n. During the key generation, encryption and decryption operations, polynomial arithmetic such as polynomial addition, subtraction and multiplication are performed. In addition, the key-generation and encryption operations require sampling of error polynomials from an error distribution (typically a discrete Gaussian.)

The ring-LWE encryption scheme is described in this way:

- In the key generation phase, two error polynomials r_1 and r_2 are sampled from the discrete Gaussian distribution. The secret key is the polynomial r_2 and the public key is the polynomial $p = r_1 - g * r_2$. After key generation, there is no use of the polynomial r_1. The polynomial g is globally known.
- In the encryption operation of a binary message vector m of length n, the message is first lifted to a ring element $\bar{m} \in R_q$ by multiplying the message bits by $q/2$. The ciphertext is computed as a pair of polynomials (c_1, c_2) where $c_1 = g * e_1 + e_2$ and $c_2 = p * e_1 + e_3 + \bar{m} \in R_q$. The encryption operation requires generation of three error polynomials e_1, e_2 and e_3.
- The decryption operation uses the private key r_2 to compute the message as $m = \mathrm{th}(c_1 * r_2 + c_2)$. The decoding function th is a simple threshold decoder that is applied coefficient-wise and is defined as

$$\mathrm{th}(x) = \begin{cases} 0 & \text{if } x \in (0, q/4) \cup (3q/4, q) \\ 1 & \text{if } x \in (q/4, 3q/4) \end{cases} \tag{1}$$

Efficiency improvements. To achieve an efficient implementation of the encryption scheme, the irreducible polynomial $f(x)$ is taken as $x^n + 1$ where

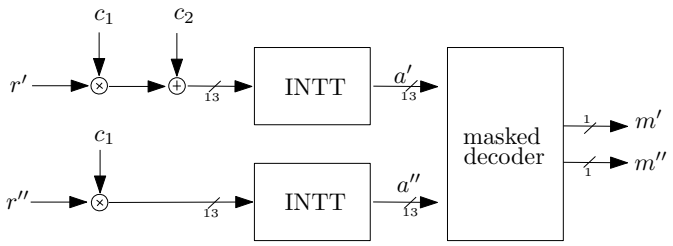


Figure 1 – General data flow of the masked ring-LWE decryption. r' and r'' are the arithmetic shares of the private key r; c_1 and c_2 are the input unmasked ciphertext; m' and m'' are the Boolean shares of the recovered plaintext.

n is a power of two, and the modulus q is chosen as a prime number satisfying $q \equiv 1 \mod 2n$ [PG14, RVM+14]. In this setting, polynomial multiplications can be efficiently performed in $O(n \log n)$ time using the Number Theoretic Transform (NTT).

Following [RVM+14], we keep the ciphertext polynomials c_1 and c_2 in the NTT domain to reduce the computation cost of the decryption operation. The decryption operation thus computes the decrypted message as

$$m = \mathrm{th}\big(\mathrm{INTT}(\tilde{c}_1 \cdot \tilde{r}_2 + \tilde{c}_2)\big). \tag{2}$$

Here the symbol $\tilde{r}$ represents the NTT of a polynomial r, and $\mathrm{INTT}(\cdot)$ represents the inverse NTT operation. The multiplication of $\tilde{c}_1 \cdot \tilde{r}_2$ is thus performed coefficient-wise (as well as the addition $\tilde{c}_1 \cdot \tilde{r}_2 + \tilde{c}_2$.) For convenience, we drop the tildes in the rest of the paper and work with c_1, c_2 and r_2 in the NTT domain. We furthermore refer to $\tilde{r}_2$ simply as r. (We recall that the INTT is a linear transformation applied to the n coefficients of $a = r \cdot c_1 + c_2$.) The decoding function th applies a threshold function to each coefficient of a as defined in Equation 1 to output n recovered message bits.

3 High-level overview

In this section, we give a high-level view of the masked ring-LWE implementation. The most natural way to split the computation of the decryption as Equation 2 is to split the secret polynomial r additively into two shares r' and r'' such that $r[i] = r'[i] + r''[i] \pmod q$ for all i. The n coefficients of r' are chosen uniformly at random in $\mathbb{Z}_q$ in each execution of the decryption.

The bulk of the computation from Equation 2 is amenable to this splitting, since by linearity of the multiplication and INTT operation, we have that $\mathrm{INTT}(r \cdot c_2 + c_1) = \mathrm{INTT}(r' \cdot c_2 + c_1) + \mathrm{INTT}(r'' \cdot c_2)$. Thus, we can split almost the entire computation from Equation 2 into two branches, as drawn in Figure 1. The first branch computes on r' to determine the polynomial

$$a' = \mathrm{INTT}(r' \cdot c_2 + c_1) \tag{3}$$

and the second branch operates on r'' to determine

$$a'' = \mathrm{INTT}(r'' \cdot c_2). \tag{4}$$

The advantage of such a high-level masking is that the operations of Equation 3 and 4 can be performed on an arithmetic processor without any particular protection against DPA. (This is because any intermediate appearing in either branch is independent of the secret r. This situation is very similar to, for example, base point blinding in elliptic curve scalar multiplication.) We can reuse an existing ring-LWE processor for these operations, and leverage the numerous optimizations carried out for this block [PG14, RVM+14].

The final threshold $\mathrm{th}(\cdot)$ operation of Equation 2 is obviously non-linear in the base field $\mathbb{F}_q$, and hence cannot be independently applied to each branch (Equation 3 and 4). There are generic approaches to mask arbitrary functions. For instance, in [BGL+14] an approach based on masked tables was used. However, these generic approaches are usually quite expensive in terms of area or randomness. In the following Section 4, we pursue another direction. We design a bespoke masked decoder that results in a very compact implementation.

4 Masked decoder

In this section we describe a very compact, probabilistic masked decoder. In the sequel, a denotes a single coefficient and (a', a'') its shares such that $a' + a'' = a \pmod q$. The decoder computes the function $\mathrm{th}(a)$ from the shares (a', a''). We also drop the symbol $\pmod q$ when obvious.

First crack. The key idea of the efficient masked decoder is that we do not need to know the exact values of the shares a' and a'' of a coefficient a in order to compute $\mathrm{th}(a)$. For example, if $0 < a' < q/4$ and $q/4 < a'' < q/2$ then $a = a' + a''$ is bounded by $q/4 < a < 3q/4$, and thus $\mathrm{th}(a) = 1$. That is, we learnt $\mathrm{th}(a)$ from only a few most significant bits from a' and a''. We can use this idea to substantially simplify the complexity of the masked th function.

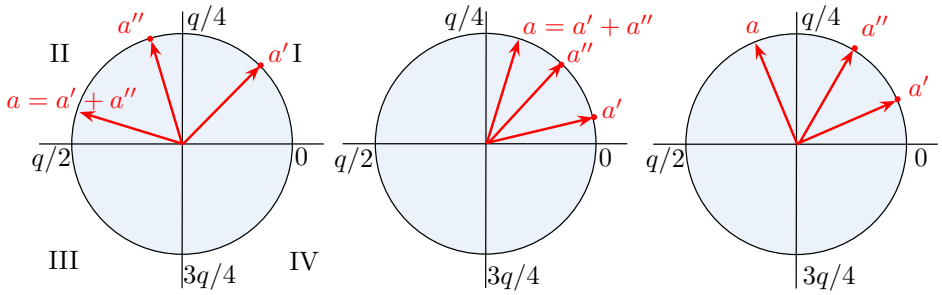


Figure 2 – Idea for the masked decoder. Elements in $\mathbb{Z}_q$ are shown in a circle. Adding two elements translates into adding their respective angles. Left: case $0 < a' < q/4$, $q/4 < a'' < q/2$, and therefore $\mathrm{th}(a) = 1$. Center and right: case $0 < a' < q/4$, $0 < a'' < q/4$, which does not allow to infer $\mathrm{th}(a)$.

4.1 Rules

Figure 2, left, illustrates the situation from the last paragraph. In this case, $0 < a' < q/4$ and $q/4 < a'' < q/2$ so obviously a can range only from $q/4$ to $3q/4$, and hence $\mathrm{th}(a) = 1$. Analogously to this rule, we can formulate 3 other rules:

- If $q/2 < a' < 3q/4$ and $3q/4 < a'' < q$ then $q/4 < a < 3q/4$ and thus $\mathrm{th}(a) = 1$.
- If $q/4 < a' < q/2$ and $q/2 < a'' < 3q/4$ then a belongs to $(0, q/4) \cup (3q/4, q)$ and thus $\mathrm{th}(a) = 0$ (quadrants I and IV, left half of the circle).
- If $3q/4 < a' < q$ and $0 < a'' < q/4$ then a belongs to $(0, q/4) \cup (3q/4, q)$ and thus $\mathrm{th}(a) = 0$.

There are 4 other rules that result from interchanging a' with a'' in the above expressions. (This follows straight from the symmetry of the additive splitting.) Essentially, with the only information of the *quadrant* of each share a' and a'' we can, in half of the cases, deduce the output of $\mathrm{th}(a)$. (For the explanation simplicity, we obviated what happens in the boundaries of the quadrant intervals. Similar conclusions hold when including them.)

What if no rule is hit? In roughly half of the cases, we can apply one of the 8 rules previously described to deduce the value of $\mathrm{th}(a)$. However, in the other half of the cases, none of the rules applies. A representative case of this event is shown in Figure 2, center and right. In both cases, $0 < a' < q/4$ and $0 < a'' < q/4$. This situation is not covered by any of the 8 rules previously described. We see that in the center sub figure $\mathrm{th}(a) = 0$ while in the right sub figure $\mathrm{th}(a) = 1$, so in this case the quadrants of each share a' and a'' do not allow us to infer $\mathrm{th}(a)$.

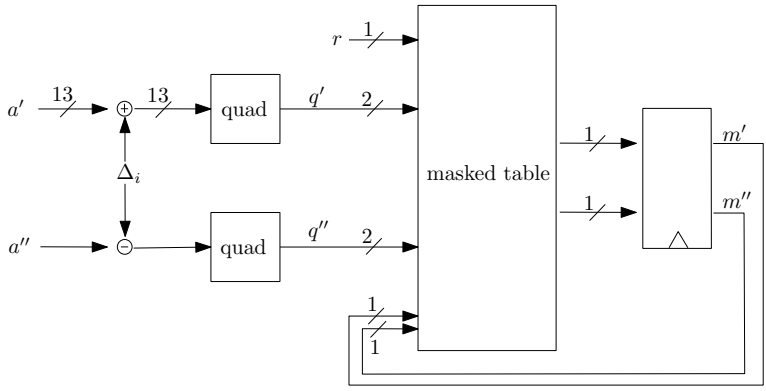


Figure 3 – The masked decoder.

The solution in this case is to refresh the splitting (a', a''), that is, update $a' \leftarrow a' + \Delta_1$ and $a'' \leftarrow a'' - \Delta_1$ for certain Δ_1. (This refreshing naturally preserves the unshared value $a = a' + a''$.) After the refreshing, the 8 rules can be checked again. If still no rule applies, the process is repeated with a different refreshing value Δ_i. Note that in each iteration of the step, roughly half of the possible values of $(a', a'') \in \mathbb{Z}_q \times \mathbb{Z}_q$ are successfully decoded, and thus the amount of pairs (a', a'') that do not get decoded shrinks exponentially with the number of iterations. In our implementation, $N = 16$ iterations produces a satisfactory result. This will be studied in detail in Section 6.1.

Optimal cooked values for Δ_i. One can determine a sequence of Δ_i values that maximizes the number of pairs successfully decoded after N iterations. We performed a first-order search for such a sequence of Δ_i values. Each Δ_i maximizes the number of successfully decoded pairs after $i - 1$ iterations. For $q = 7681$ the sequence of Δ_i shown in Section 9 on page 145 was found.

Architecture. The hardware architecture for the masked decoder follows from the previous working principle description. Our implementation is shown in Figure 3. From left to right, we see the first refreshing step by the constants Δ_i. The constants Δ_i vary from iteration to iteration. After the refreshing step, the quadrant function is applied to each share a', a''. This quadrant function outputs x if a belongs to the x-th quadrant, and thus the output consists of 2 bits. These blocks are essentially 13-bit comparators, and thus relatively inexpensive in logic.[1] The subsequent rule checking on (q', q'') is performed by a masked table lookup that is described in the following section. The whole process is repeated $N = 16$ iterations, and this

[1] Note that in the special case that q is a prime close to a power of two the construction of the quadrant block can be further simplified.

number of iterations stays fixed even if the decoding is successful after the few first iterations.

4.2 Masked table lookup

The final step in the masked decoder is a masked table lookup. This table implements the rules described in Section 4.1, and essentially maps the output of each quadrant q'_i and q''_i (2 bits each) after the i-the iteration ($i \in [1, N]$) to a (Boolean) masked output bit value (m'_i, m''_i). In our specific implementation, we have other inputs: the result of the decoding from the previous iteration (m'_{i-1}, m''_{i-1}) and an extra randomness bit r (fresh at each of the N iterations for each of the n coefficients).

This is a well-studied problem that arises in other situations (for instance, when masking the sbox lookup in a typical block cipher) and there are plenty of approaches here to implement such masked table lookup. We opted for the approach of masked tables as in [Tri13]. We set $m'_i \leftarrow r$ and we compute $m''_i \leftarrow f(r, q'_i, q''_i, m'_{i-1}, m''_{i-1})$. The function f essentially bypasses the previous decoded value when no rule applies to q'_i, q''_i by setting the output m''_i to $r + m'_{i-1} + m''_{i-1}$ (refreshing the content of the output registers). If a rule applies to q'_i, q''_i, it sets the output m''_i accordingly. By doing this, we can register always the output of this table and no control logic to enable such output register is needed (it is implicitly integrated into this masked table.) This is the reason why the table sees also the previous decoded value m'_{i-1} and m''_{i-1}.

The usual precautions are applied when implementing f. For our target FPGA platform, we carefully split the 7-bit input to 1-bit output function f into a balanced tree of 4-bit input LUTs, in such a way that any intermediate input or output of LUTs does not leak in the first order. Note that here we are assuming that each LUT is an atomic operation. If stronger security guarantees are needed, other approaches to implement such function f should be followed. When implemented in an ASIC, it may be preferable to store this masked table in ROM (since the contents of the table are immutable and the size is small.)

The output of this table is (Boolean) masked, and thus no unmasked value lives within the implementation. This is suited for consumption of a masked AES module (say) after some preprocessing as will be detailed later. We stress that we use masked tables on the *output* of the quadrants. This is the key for our reduced area requirements, as will be explained in the next Section 5.

5 Implementation results

We implemented the fully masked ring-LWE decryption system with the parameter set $(n, q, s) = (256, 7681, 11.32)$ first introduced in [GFS+12], corresponding to a medium-term security level. The target platform is a Xilinx Virtex-II xc2vp7 FPGA. The HDL files were synthesized within Xilinx ISE v8.2 with optimization settings set to balanced and `KEEP HIERARCHY` flag when appropriate to prevent optimization of security-critical components. We base our arithmetic processor on the design from [RVM+14].

5.1 Area

In our case, a single arithmetic coprocessor performs serially the computations of Equation 3 and then that of Equation 4. This incurs in a very slight area overhead (only the control microcode is slightly modified, plus the masked decoder), at the obvious cost of an increased execution time. In comparison to the unprotected version, our protected decryption scheme consumes more memory as now we store two shares r' and r'' of the secret polynomial r, and the two output polynomials a' and a'' from the two INTT operations.

In Table 1, we can see that the proposed masking of the ring-LWE architecture incurs an additional area overhead of only 301 LUTs and 129 FFs in comparison to the unprotected version. [2] This additional area cost is mostly due to a pair of masked decoders. Due to its low area overhead, we chose to keep two masked decoders in parallel, decoding two coefficients simultaneously. (This nicely fits with the memory organization of the arithmetic coprocessor, since it fits two 13-bit coefficients in each memory word.) Thus, we use two addition and subtraction circuits for the refreshing with Δ_i (accounting for 160 LUTs) and two masked tables (90 LUTs in total.)

We note that we could straightforward reduce the additional area cost by reusing the 13-bit addition and subtraction circuits present in the arithmetic coprocessor. Since during a decoding operation, the arithmetic coprocessor remains idle, reusing of the addition and subtraction circuits do not cause any increase in the cycle count. For simplicity, we did not implement this approach.

5.2 Cycle count

The cycle count for our approach is decomposed in the computation of Equation 3, Equation 4 and the masked decoder. Equation 3 takes 2840 cycles (one unprotected

[2] Note that these results are not directly comparable with [RVM+14], since the latter were obtained from a more advanced Virtex-6 FPGA, which has 6-bit input LUTs and superior routing mechanisms in comparison to our target FPGA.

Implementation Algorithm	LUTs/FFs/DSPs	Freq (MHz)	Cycles/Time(μs) Decryption
Unprotected RLWE	1713/830/1	120	2.8k/23.5
Protected RLWE	2014/959/1	100	7.5k/75.2

Table 1 – Performance and Comparison on Xilinx Virtex-II xc2vp7 FPGA.

ring-LWE decryption), Equation 4 takes 2590 cycles, slightly less than Equation 3 since there is no addition present in the second branch.

The two-way parallel masked decoder takes $\frac{1}{2} \times n \times N + \epsilon$ cycles to decode all the coefficients into message bits. In our case with $n = 256$, $N = 16$ the masked decoder takes 2048 cycles. Thus in total, a masked decryption operation requires 7478 cycles. The arithmetic coprocessor and the masked decoder run in constant time and constant flow.

5.3 Comparison with an elliptic-curve cryptosystem

We compare our protected decryption scheme with the unprotected high-speed elliptic curve scalar multiplier architecture proposed by Rebeiro et al. in [RRM12]. The architecture for the field GF(2^{233}) consumes 23 147 LUTs and computes an unprotected scalar multiplication in 12.5μs on a more advanced Virtex-4 FPGA. Thus the scalar multiplier has an area $\times$ time product of approximately 289 337. Our protected ring-LWE decryption (for a similar security) achieves an area $\times$ time product of approximately 151 452 on a Virtex-2 FPGA; thus achieving at least 1.9 times better figure of merit.

5.4 Trade-offs

The previous figures are subject to trade-offs. If smaller latency is desired instead of a compact implementation, two coprocessors can perform the two computations of Equation 3 and 4 in parallel. Trade-offs also apply to the masked decoder, and the parallelization could be extended easily to reduce latency in this stage. Since the BRAMs present in the Xilinx FPGAs support reading of multiple consecutive words, we could keep more pairs of masked decoders in parallel and reduce the number of cycles. Another alternative is to keep the masked decoder in pipeline with the polynomial arithmetic block. Such type of setting is suitable for systems where many decryption operations are performed in a chain. While the masked decoder works on the coefficients of a previous computation, the polynomial arithmetic unit processes new ciphertexts. Since the masked decoder is faster than the polynomial arithmetic unit, the cycle count of the masked decoder is not an overhead in such

type of setting. But of course, in this situation we could not reuse the arithmetic circuitry of the arithmetic coprocessor for the refreshing operation of the masked decoder.

5.5 Maximum frequency

We note that the arithmetic coprocessor is a very optimized unit with a complex pipeline organization. We thus insert two pipeline stages in the masked decoder to match the maximum frequency of the whole system to that of the arithmetic coprocessor. In this way, the design can run up to almost 100 MHz. The critical path is inside the arithmetic multiplier.

6 Discussion

6.1 Error rates

Cryptosystems based on ring-LWE are inherently probabilistic. This means that there is a non-zero probability that the recovered plaintext after ring-LWE decryption is not exactly the plaintext before encryption. In our case, due to the probabilistic nature of our masked decoder approach, there is a second source of noise. Since the number of iterations of the masked decoder is finite, there are some pair values (a', a'') that will not get decoded within the fixed finite number of iterations. In this section, we first explain the error rate of the probabilistic decoding in isolation, and then we switch to the global system error rate and point out strategies to mitigate it.

Errors due to the probabilistic decoding. In this section, we assume that the plaintext bit is 1 and the unmasked input a to the masked decoder is in $(q/4, 3q/4)$. The additional error due to the probabilistic masked decoder is the probability p_e that (a', a'') does not get successfully decoded. Let us write $p_s = 1 - p_e$.

This probability p_s is influenced by two distributions. We have that

$$p_s = \sum \Pr[\text{successful decode}|a] \cdot \Pr[a] \tag{5}$$

where the sum is taken over $a \in (q/4, 3q/4)$. On the one hand, $\Pr[\text{successful decode}|a]$ is the probability that the decoder successfully decodes a. On the other, $\Pr[a]$ is the probability with which a takes various values in $(q/4, 3q/4)$.

The distribution of the decoder success probability $\Pr[\text{successful decode}|a]$ as a function of the unshared input value a to the decoder can be easily

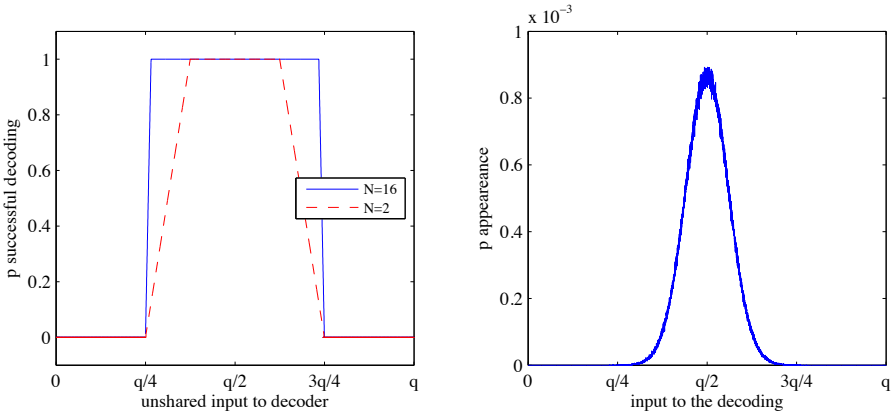


Figure 4 – Left: empirical success distribution for the masked decoder. Right: Distribution of a when plaintext is 1.

computed by averaging over all possible pairs (a', a'') such that $a' + a'' = a$. Since for any given value of a, its shares a' or a'' are (individually) equiprobable, we compute $\Pr[\text{successful decode}|a]$ as $\Pr[\text{successful decode}|a] = \frac{1}{q}\sum_{a'+a''=a} \Pr[\text{successful decode of } (a', a'')]$.

The distribution $\Pr[\text{successful decode}|a]$ is shown in Figure 4, left. We see that the decoder performs best when $a \approx q/2$, in which case all possible inputs get decoded correctly. Only when the input value a approaches $q/4$ or $3q/4$, the performance degrades. When using a larger number of iterations $N = 16$ this effect is less pronounced when compared to $N = 2$ iterations, as Figure 4 shows.

On the other hand, it is easy to see that not all unshared inputs a to the decoder are equally likely. By the construction of the ring-LWE decryption function, the unshared input to the decoder a is either centered around $q/2$ (resp. 0) when the message bit is 1 (resp. 0). This distribution $\Pr[a]$ is plotted in Figure 4, right.

These two observations combined produce a nice interaction between the *prior* distribution $\Pr[a]$ of a (given by the ring-LWE decryption) and the success distribution of the masked decoder $\Pr[\text{successful decode}|a]$ as in Equation 5. Namely, values of a that are difficult to decode (those with low $\Pr[\text{successful decode}|a]$) are quite unlikely to appear as input to the masked decoder (their $\Pr[a]$ is also low). This positive interaction keeps the global error rate of the system quite low. This is precisely quantified in the next paragraph.

Global error rate and number of iterations. We performed simulations to estimate the global error rate and determine the required number of iterations N

Iterations	p_g $[\times 10^{-5}]$	p_g/p_{baseline}
N =2	332.24	91.41
3	178.44	49.09
4	25.36	6.97
5	20.77	5.71
6	16.22	4.46
8	6.97	1.91
16	4.32	1.19
24	4.06	1.11
30	3.87	1.06

Figure 5 – Global error rates with the probabilistic decoder.

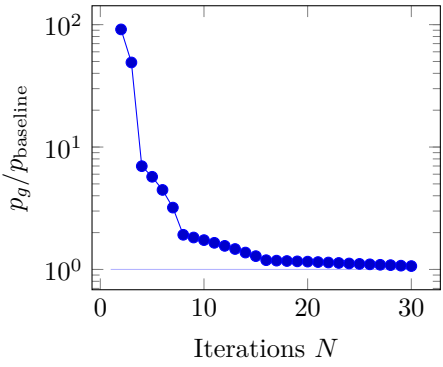


Figure 6 – Evolution of the ratio p_g/p_{baseline} as the number of iterations N grows.

in our design. Over 10^6 bits, the average error per bit using a deterministic decoder was $p_{\text{baseline}} = 3.634375 \times 10^{-5}$. This is a baseline error intrinsic to the ring-LWE construction. When we plug in the probabilistic decoder, the global, end-to-end, error rate per bit p_g increases. (We have $p_g = p_{\text{baseline}} + p_e$.) In Figure 5, we can find the global error rate for different values of the number of iterations N of the decoding. At $N = 3$, for instance, the error rate is $p_g = 1.7844 \times 10^{-3}$, which is ≈ 49 times larger than p_{baseline}. As already hinted, the error rate quickly decreases with N (roughly exponentially, as can be see in Figure 6). In our design, we set $N = 16$ (we iterate 16 times per coefficient) as a balanced tradeoff between cycle count and error rate. The impact of the masked probabilistic decoder on the global error rate is quite low, adding less than 20% to the intrinsic error rate when compared to a deterministic decoder, as it can be see in Figure 5.

6.2 Comparison with other decoding strategies

We are only aware of a similar masked decoder, the one presented in [BGL+14]. There the authors resort to a generic masking method, namely masked tables, to perform the decoding. Translating the ideas of [BGL+14] in our context, we would need two tables of 2^{13} bits (one of them random). For a smaller group $\mathbb{Z}_d$ with $d = 257$ the authors report an utilization of 1331 slices on a Virtex 6 FPGA. While the results in slices are not directly comparable with ours, we point out that the size of the masked table following the approach of [BGL+14] grows linearly in the group size q, while for our solution the size of the masked table stays constant (independent of q), and the quadrant blocks grow only logarithmically in q. The

cycle count, however, is larger in our solution. The critical observation of our masked decoder is that we can *compress* the input coefficient shares a' and a'' to a mere two bit per share (the output of each quadrant) and then perform the decoding based on the information of the two quadrants (4 bits.)

6.3 Post-processing

Albeit the computation from Equation (2) is commonly referred as the "ring-LWE decryption", the decryption process should include a post-processing on the recovered message m. This post-processing consists of error correction and padding verification.

Linear codes with masking. One approach to deal with the probabilistic nature of the ring-LWE decryption system is to use forward error correcting codes (FEC). The message prior to encryption is encoded using a FEC and the resulting composite is ring-LWE encrypted. The output of the ring-LWE decryption should be corrected for errors, preferably in the masked domain. For syndrome decoding of linear codes, this can easily be done by masking the syndrome table.

Padding schemes. As presented, the ring-LWE system is malleable. CCA security can be achieved with a padding mechanism. The Fujisaki-Okamoto [FO13] padding scheme is known to work with ring-LWE [Pei14]. This padding scheme makes use of standard symmetric cryptographic constructions whose masked implementations are well studied. We point out that key-encapsulation mechanisms may result in a more compact and simpler implementation.

6.4 Extension to higher-order security

We point out that the approach laid out in Section 3 scales quite well with the security order. To achieve security at level $d + 1$, one would need to split the computation of Equation 2 into d branches analogously to Equation 3. The masked decoder can follow the same principles with the appropriate modifications. The complexity of this decoder obviously grows. Generic approaches to perform this computation have been discussed in [Cor14, BGN$^+$14, RBN$^+$15].

7 Evaluation

For the purposes of a side-channel evaluation, we implemented the full design on a SASEBO G board. The design was clocked at 18.75 MHz and the power consumption was sampled at 500 MS/s. This platform is very low noise.

We provide a very advantageous setting for the adversary: we assume that the evaluator knows the details about the implementation (for example, pipeline stages). In addition, we assume that while guessing a subkey, the adversary knows the rest of the key. These assumptions allow to comfortably place predictions on intermediates arbitrarily deep into the computation. While this may represent a very powerful attacker and somewhat unrealistic, the algebraic structure of such cryptosystem may help the attacker to predict deep intermediates with relatively low effort. We refer the reader to the extended version of this paper for an attack on half-masked ring-LWE decryption that uses these ideas. This stresses the necessity of masking the decoding function entirely.

The evaluation methodology to test if the masking is sound is as follows. We first proceed with first-order key-recovery attacks when the randomness source (PRNG) is switched off. We demonstrate that in that situation the attacks are successful, indicating that the setup and procedure is sound. Then we switch on the PRNG and repeat the attacks. If the masking is sound, the first-order attacks shall not succeed. In addition, we perform second-order attacks to confirm that the previous first-order analyses were carried out with enough traces.

We test 4 different points which covers all the relevant parts of the computation. The targets are the first 13-bit coefficient of $r' \cdot c_1 + c_2$, the first 13-bit coefficient of $r'' \cdot c_1$, the first input coefficient to the shared decoder and the first output bit. We modeled the power consumption of a register as the Hamming distance between two consecutive values held in the register, and used Pearson's correlation coefficient to compare predictions with measurements [BCO04].

7.1 PRNG off

We first begin the experiments when the PRNG is off. That is, the sharing of r into r' and r'' on each execution is deterministic. This would not happen in practice, as an active PRNG would randomize the representation of r in each execution. In our setting, this would mean that the masking is switched off.

In Figure 7 we draw the result of correlating against the 4 intermediates with 10 000 traces. On top, we draw a mean trace for orientation. The correlation values are, from top to bottom, 0.25, 0.3, 0.27 and 0.21, respectively. This means that the attacks are successful, and confirms the soundness of our setting. In Figure 8 we can see the evolution of the correlation coefficient as the number of traces increases

for the first two intermediates. We can see that starting from hundred traces the attack is successful. Similar behavior was observed for other intermediates.

7.2 PRNG on

In Figure 9 we draw the result of the previous analysis when the masks are switched on. This corresponds to the situation that an adversary would face in reality. We can see that the correct key guess is no longer distinguishable, even when using 10 000 traces. We repeated the same experiments for other intermediates and other intermediate positions with identical results.

7.3 Second-order attacks

To confirm that we used enough traces in our previous analyses, we perform here second-order attacks on the masked implementation with the PRNG on. We will focus on the masked decoder. In Figure 10 we draw on top a mean curve in the region of 7 400 to 7 700 cycles, corresponding to the end of the masked decoding. We target one output bit of the decoding: $m[254]$.

In Figure 10 we first begin by correlating against masks and masked values. This is a test scenario, since for this attack we need to know the masks, something that would not happen in a real deployment. Correlation with masks or masked value yield high correlation as expected ($\rho = 0.32$ and $\rho = 0.34$, respectively). In contrast, when correlating against the unshared value (in light blue), the correlation coefficient does not traverse the confidence interval for $\rho = 0$. This indicates that the masking is effective. We can repeat the same attack against centered and squared traces [CJRR99, PRB09]. This is effectively a second-order attack, and is expected to work. It is shown in magenta in Figure 10, and we can see that the attack succeeds. Using the centered absolute value to pre-process traces also works as expected, as shown in yellow.

In Figure 11 we can see the evolution as a function of the number of traces. We can see that starting from ≈ 2000 measurements this second-order attack is successful. This confirms that the first-order attacks of Section 7.2 were carried out with enough traces, since a second-order attack is already successful starting from ≈ 2000 measurements.

We remark that the relatively low number of traces required for the second-order attack is due to the very friendly scenario for the evaluator. The platform is low noise and no other countermeasure except than masking was implemented. In practice, masking needs a source of noise to be effective, and consequently the higher-order attacks would be harder to mount, requiring more traces [CJRR99] and more computation [RGV12].

7.4 Horizontal DPA attacks

During the decoder operation, the input coefficients are refreshed $N - 1 = 15$ times with publicly known offsets Δ_i. The device thus handles consecutively the values a', $a' + \Delta_1$, ..., $a' + \Delta_1 + \ldots + \Delta_{15}$. This may enable a horizontal DPA attack [PdHL09] during the operation: the adversary may collect a single trace, split it into 16 chunks and then perform a DPA on these 16 chunks to recover the mask a'. Once the masks from all traces are discovered, a first-order, vertical DPA applies.

There are two factors that mitigate this threat. First, we note the adversary is given a very limited number of traces to recover each mask (namely, $N = 16$). Secondly, this attack can be easily prevented by shuffling the public coefficients Δ_i. This randomizes the order of execution of each refreshing with Δ_i, and thus the exposure to horizontal DPA attacks is minimized.

8 Conclusion

In this paper we described a practical side-channel protected implementation of the lattice-based ring-LWE asymmetric decryption. Our solution is based on the sound principles of masking and incurs in a manageable overhead (in cycles and area). A key component of our solution is a bespoke masked decoder. Our implementation performs the entire ring-LWE decryption computation in the masked domain.

Acknowledgements. The authors would like to thank the CHES 2015 reviewers for their valuable comments. This work has been supported in part by the European Commission through the ICT programme under contracts H2020-ICT-645622 PQCRYPTO, H2020-ICT-644209 HEAT and FP7-ICT-2013-10-SEP-210076296 PRACTICE; by the Research Council KU Leuven TENSE (GOA/11/007); by the Flemish Government FWO G.0550.12N, G.00130.13N and G.0876.14N; and by the Hercules Foundation AKUL/11/19. Oscar Reparaz is funded by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO). Sujoy Sinha Roy was supported by Erasmus Mundus PhD Scholarship.

References

[BBD08] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. *Post Quantum Cryptography*. Springer, 1st edition, 2008.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems* -

CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings, volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2004.

[BGL+14] Hai Brenner, Lubos Gaspar, Gaëtan Leurent, Alon Rosen, and François-Xavier Standaert. FPGA Implementations of SPRING - And Their Countermeasures against Side-Channel Attacks. In *CHES*, volume 8731 of *LNCS*, pages 414–432. Springer, 2014.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BLLN13] Joppe W Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In *Cryptography and Coding*, volume 8308 of *LNCS*, pages 45–64. Springer, 2013.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[Cor14] Jean-Sébastien Coron. Higher Order Masking of Look-Up Tables. In Phong Q Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[DDLL13] Léo Ducas, Alain Durmus, Tancréde Lepoint, and Vadim Lyubashevsky. Lattice Signatures and Bimodal Gaussians. In *CRYPTO*, volume 8042 of *LNCS*, pages 40–56. Springer, 2013.

[FO13] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *Journal of Cryptology*, 26(1):80–101, 2013.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012.

[GFS+12] Norman Göttert, Thomas Feller, Michael Schneider, Johannes Buchmann, and Sorin Huss. On the Design of Hardware Building Blocks for Modern Lattice-Based Encryption Schemes. In *CHES*, volume 7428 of *LNCS*, pages 512–529. Springer, 2012.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. In *EUROCRYPT*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010.

[PdHL09] Jing Pan, J I den Hartog, and Jiqiang Lu. You Cannot Hide behind the Mask: Power Analysis on a Provably Secure \emphS-Box Implementation. In Heung Youl Youm and Moti Yung, editors, *Information Security Applications, 10th International Workshop, WISA 2009, Busan, Korea, August 25-27, 2009, Revised Selected Papers*, volume 5932 of *Lecture Notes in Computer Science*, pages 178–192. Springer, 2009.

[Pei14] Chris Peikert. Lattice Cryptography for the Internet. In *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, pages 197–219, 2014.

[PG14] Thomas Pöppelmann and Tim Güneysu. Towards Practical Lattice-Based Public-Key Encryption on Reconfigurable Hardware. In *Selected Areas in Cryptography – SAC 2013*, volume 8282 of *LNCS*, pages 68–85. Springer, 2014.

[PRB09] Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[Reg05] Oded Regev. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 84–93, New York, NY, USA, 2005. ACM.

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

[RRM12] Chester Rebeiro, Sujoy Sinha Roy, and Debdeep Mukhopadhyay. Pushing the Limits of High-Speed GF(2ˆm) Elliptic Curve Scalar Multiplication on FPGAs. In *CHES*, volume 7428 of *LNCS*, pages 494–511. Springer, 2012.

[RRVV14] Sujoy Sinha Roy, Oscar Reparaz, Frederik Vercauteren, and Ingrid Verbauwhede. Compact and Side Channel Secure Discrete Gaussian Sampling. *IACR Cryptology ePrint Archive*, 2014:591, 2014.

[RVM+14] Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact Ring-LWE Cryptoprocessor. In *CHES*, volume 8731 of *LNCS*, pages 371–391. Springer, 2014.

[THM+07] Michael Tunstall, Neil Hanley, Robert P McEvoy, Claire Whelan, Colin C Murphy, and William P Marnane. Correlation Power Analysis of Large Word Sizes. *IET Irish Signals and Systems Conference (ISSC) 2007*, sep 2007.

[Tri13] E V Trichina. Table lookup operation on masked data, 2013. US Patent 8,422,668.

9 Optimal values of Δ_i for $q = 7681$

$$\begin{aligned}
\Delta(i) = (&960, 1440, 480, 1680, 240, 720, 1200, 1800, && (6)\\
&120, 360, 600, 840, 1080, 1320, 1560, 1860, && (7)\\
&60, 180, 300, 420, 540, 660, 780, 900, 1020, && (8)\\
&1140, 1260, 1380, 1500, 1620, 1740, 1890, && (9)\\
&30, 90, 150, 210, 270, 330, 390, 450, 510, && (10)\\
&570, 630, 690, 750, 810, 870, 930, 990, 1050, && (11)\\
&1110, 1170, 1230) && (12)
\end{aligned}$$

These values were found by exhaustive first-order search. The value Δ_i is chosen so that it maximizes the number of pairs that get decoded after i iterations.

10 Attack on half-masked variant

In this section, we analyze the security of a masked ring-LWE variant where the intermediates just before decoding are unmasked, and the decoding is performed in the unmasked domain. This alternative is definitely cheaper than full masking. In the following, we provide evidence to show that this clearly does not provide enough security in our case.

(A seemingly similar situation appears in [BGL+14]. However, there are important differences—namely it is not possible to choose ciphertexts. In the following, we are not analyzing the variant of [BGL+14] but only the half-masked ring-LWE.)

A common argument is that after key-diffusion is complete, prediction of the intermediates is not possible and hence standard DPA attacks to the half-masked ring-LWE do not apply. We will see that this is not strictly true, if the attacker can choose ciphertexts.

Assume that the coefficients of the polynomial $a = \mathrm{INTT}(r \cdot c_1 + c_2)$ appear unmasked in the implementation. Let the adversary collect measurements with chosen ciphertext. The ciphertext c_1 has the following structure: all the coefficients fixed except $c_1[0]$ randomly varying. The ciphertext c_2 has the same structure. Then observe that due to linearity of the INTT operation, $a[0]$ can be written as $a[0] = \alpha(r[0] \cdot c_1[0] + c_2[0]) + \beta$, where

- α is a public constant determined by the INTT transformation.
- β is a secret constant that is a function of the other (unknown) key coefficients $r[1], \ldots, r[255]$. Note that by construction β is constant within the set of collected traces.

Thus, an attacker can perform a DPA attack targeting the intermediate $a[0]$ and placing predictions on $(r[0], \beta)$. The adversary recovers $r[0]$ and proceeds to recover other key coefficients. We have verified this attack in simulations, even when using $\mathrm{th}(a[i])$ as intermediate.

(It may seem that the high number of hypotheses, 2^{26} may produce a cumbersome attack. However, one can apply techniques of partial correlation [BCO04] to alleviate the computational effort of DPA on large word sizes [THM+07]. And we have experimented that in practice it makes sense to first recover $r[0]$ (this is easier due to larger non-linearity of the modular multiplication) and then β (which may be harder due to the low non-linearity of the modular addition), splitting the 2^{26} effort in two 2^{13} steps.)

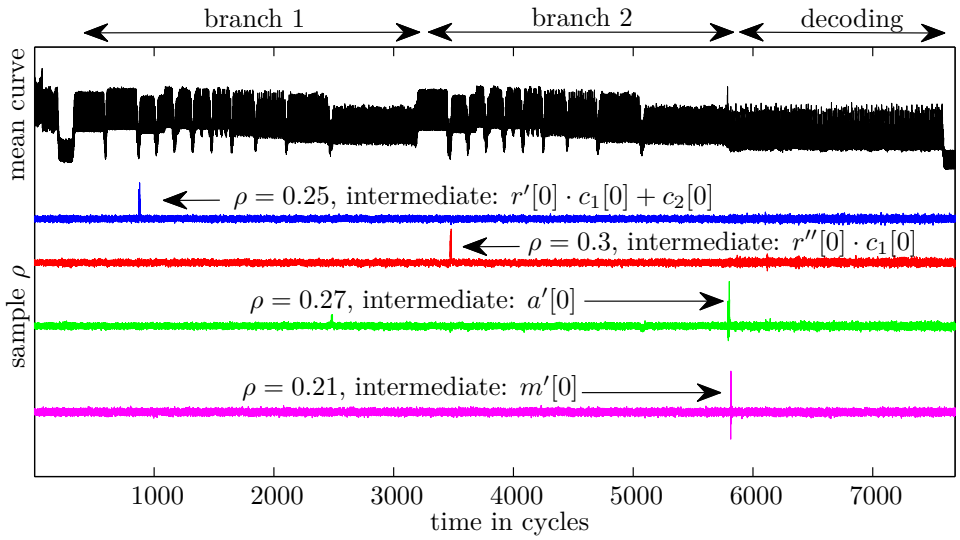


Figure 7 – PRNG off. On top, black, one power consumption trace. The different computational stages can be distinguished: first branch, second branch and decoding. Next, in blue, the correlation trace for the value $r'[0] \cdot c_1[0] + c_2[0]$. The correlation achieves a maximum value of $\rho = 0.25$. Below, in red, correlation for $r'' \cdot c_1$ (max $\rho \approx 0.3$); in green: correlation for the input of the masked decoder $a'[0]$. At the bottom: correlation with one message bit $m'[0]$.

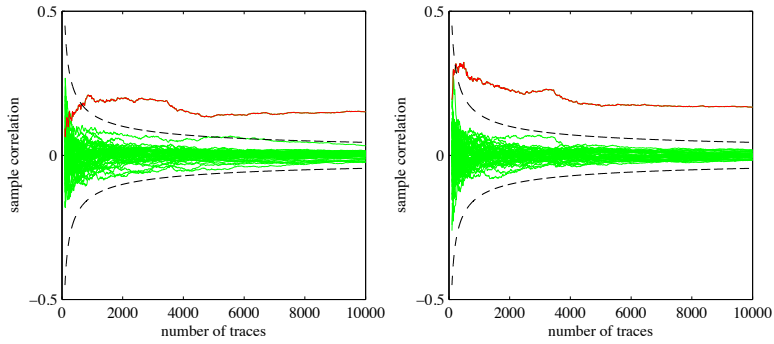


Figure 8 – PRNG off. Evolution of the correlation coefficient as the number of traces increases for the intermediates $r'[0] \cdot c_1[0] + c_2[0]$ (left) and $r''[0] \cdot c_1[0]$ (right). Correct subkey guess in red, all other guesses in green. A 99.99 % confidence interval for $\rho = 0$ is plotted in black discontinuous line. We can see that starting from hundred measurements the attacks are successful.

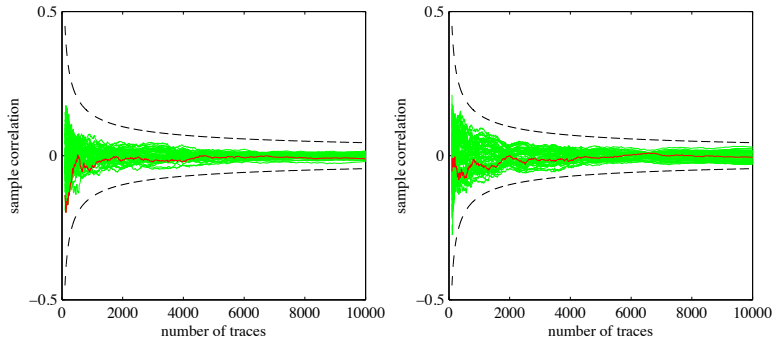


Figure 9 – Analogous to Figure 8, but with PRNG on. The correct subkey is no longer identifiable. This is expected and means that the masking is effective.

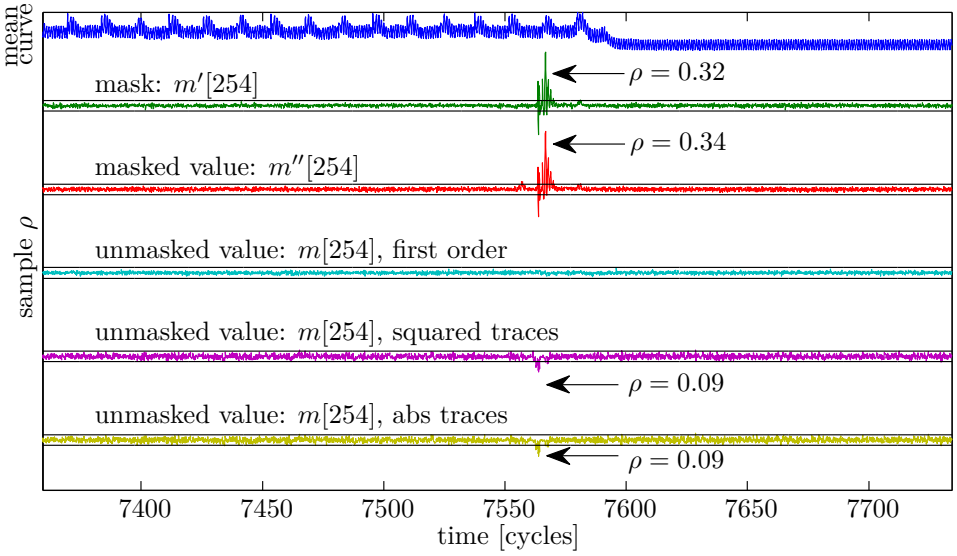


Figure 10 – Correlation traces for intermediates within the shared decoder. On top, a power measurement trace showing the last 15 decodings. Below, correlation traces. The first two (masks and masked values) assume that the adversary knows the masks. The third one, in light blue, is a first-order attack without knowing the attack, and is unsuccessful. In contrast, the second-order attack against the same intermediate is successful, as the traces in magenta and yellow show.

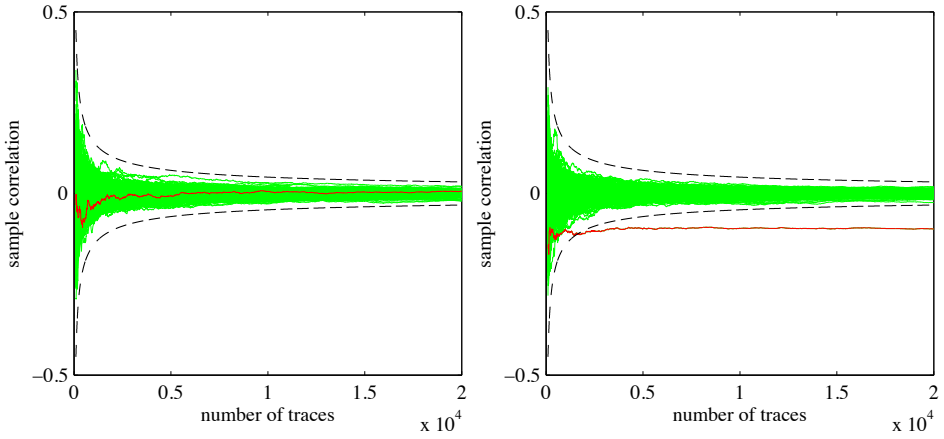


Figure 11 – Left: correlation as the number of traces increases for the first-order attack (PRNG on), around clock cycle 7560. Right: correlation for the second-order attack with masks on. The attack begins to be successful with 2 000 measurements.

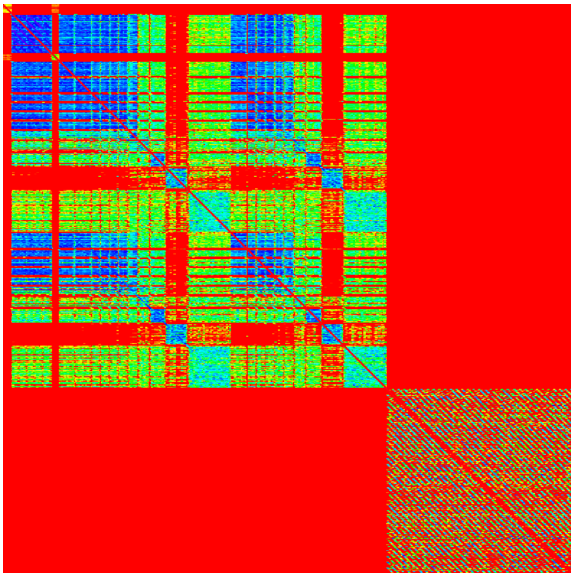


Figure 12 – Crosscorrelation trace. The x and y axes represent time, flowing from the upper left hand side corner to the lower right. The entire figure spans 7500 cycles (as Figure 7). It is possible to distinguish the two branch computations (including its components) and the decoding. Colors enhanced to improve contrast.

Publication

Consolidating masking schemes

Publication Data

Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

Contribution

Principal author, except for the appendix.

Consolidating Masking Schemes

Oscar Reparaz, Begül Bilgin,
Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede
firstname.lastname@esat.kuleuven.be

KU Leuven ESAT/COSIC and iMinds, Belgium

Abstract. In this paper we investigate relations between several masking schemes. We show that the Ishai–Sahai–Wagner private circuits construction is closely related to Threshold Implementations and the Trichina gate. The implications of this observation are manifold. We point out a higher-order weakness in higher-order Threshold Implementations, suggest a mitigation and provide new sharings that use a lower number of input shares. **Keywords:** Masking, Private Circuits, Ishai–Sahai–Wagner, Threshold Implementations, Trichina gate, higher-order DPA.

1 Introduction

Side-channel cryptanalysis allows to break implementations of mathematically secure cryptographic algorithms running on embedded devices. Shortly after the introduction of a particularly powerful branch of side-channel attacks, namely Differential Power Analysis (DPA) by Kocher et al. [KJJ99], different countermeasures were proposed. An especially popular countermeasure used today is masking, introduced in [CJRR99, GP99], mainly due to its theoretical soundness. Contrary to other heuristic, ad-hoc approaches, masking carries a proof of security [CJRR99]. A d^{th}-order secure masking works by splitting every sensitive variable (i.e. that depends on the key) into s shares, such that an adversary probing at most d values during the computation gets no information about any sensitive variable. This adversarial model is relevant in practice since the adversary is not weaker than a d^{th}-order DPA attack [DDF14, Cor14]. The advantage of a properly masked implementation is that it forces the adversary to use higher-order DPA attacks in order to break it. Higher-order DPA attacks are substantially harder to launch, both in terms of data complexity and computational resources [CJRR99, Mes00, RGV12]. Masking, however, comes with a cost. Performing operations in the masked domain increases the computational requirements on the target platform (area, time and randomness, among others), thus in practice, it is crucial to design countermeasures that have a limited cost impact.

In this paper, we focus on Boolean masking, i.e. the intermediates are split additively in a given finite field. The difficulty is then reduced to masking functions that are not linear with respect to addition.

1.1 Related works

There have been several efforts for constructing masking schemes —algorithms to compute on masked data. Some early development came from practitioners which produced designs mostly oriented to fit in real-world constraints: Trichina presents in [Tri03] a masked AND gate resistant to first-order DPA attacks (first-order secure masking).

A generic algorithm for the masked AND computation at any security level is given by Ishai, Sahai and Wagner (here ISW) in [ISW03], together with a convenient theoretical framework to prove the security of such a scheme. It is, however, well known that early theoretical concepts of masking schemes rely on assumptions that do not necessarily hold in practice. This is true for both the hardware and the software side. A common problem for the latter is that Hamming distance leakage, which is typically visible in memory-element transitions, may invalidate the assumption that leakages from each share are independent [BGG$^+$14]. For the former, glitches (in static CMOS, a spurious transition of nodes in a combinational circuit within one clock cycle, resulting from different arrival times of the input signals) were shown to be a source of exploitable leakage [MPG05, MS06], enabling successful DPA attacks against theoretically sound masked implementations due to unsatisfactory leakage modeling.

The mitigation of glitches is a well-studied problem in digital design, since they are not only inconvenient from a security point of view. Glitches are useless transitions that consume extra energy and thus digital designers tend to minimize them to achieve low-power and high-speed circuits. There are strategies to reduce glitches (e.g., balancing the path delay using combinational tree-like structures) or fully eliminate them (e.g. using dynamic logic styles, such as Domino or dynamic differential such as SABL [TAV02] or WDDL [TV04]).

Alternatively, a specific strand of masking schemes, namely Threshold Implementations (TIs), were introduced in [NRR06] to address the aforementioned model limitations. TIs are designed to deal with non-idealities in hardware (glitches) at a higher level of abstraction, and can provide strong security guarantees that may be relevant in practice. While ISW requires first to decompose a circuit into (exclusively) AND, XOR and NOT gates and then masking those, TI has the advantage that any function can be shared directly, which typically results in more compact designs. Recently TIs were extended to provide not only first-order but also higher-order security [BGN$^+$14b].

1.2 Our contribution

The discussion provided in this paper is threefold. First, we point out the similarities and differences between ISW, TI and the Trichina gate when the function to mask is an AND gate. We gain deeper understanding about masking schemes from these relations and use it to provide a generalized masking scheme (Section 3).

In the second part of the paper, we show how this generalization is mutually beneficial to all three masking schemes mentioned above. We show a weakness in the recently proposed higher-order extension of TI and suggest a fix using ideas from the generalized scheme in Section 4. In addition, we discuss how ISW and the Trichina masked AND-gate can be implemented securely in logic styles that glitch.

Finally, we focus on constructive applications. In Section 5.1, we discuss under which conditions a TI function provides security against d^{th}-order attacks using only $d+1$ shares instead of the usual $td+1$ bound. We end the paper by describing how ideas from TI could be inherited in software-oriented schemes to provide security in a distance-based leakage model (Section 5.2).

2 Preliminaries

We begin with standard definitions and descriptions of the masking schemes that we consider. Lower-case characters refer to elements in a field with characteristic two. An element a is split into s shares a_i, where $i \in 1, 2, \ldots, s$ by means of Boolean masking. Namely, without loss of generality $s-1$ random shares $a_1, \ldots, a_{s-1}$ are drawn from the uniform distribution, then a_s is calculated such that $a = \bigoplus_s a_i$. Bold characters refer to a valid shared vector $\mathbf{a} = a_1, \ldots, a_s$. We use the term s-share representation (s-sharing) of a to emphasize the number of shares. Note that the sharing $\mathbf{a}$ generated as detailed above is uniform [BGN+14a]. Moreover, the sharing $\mathbf{a_{\bar{i}}} = a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_s$ is independent of the unshared value a for any choice of i. It is hence an (s, s) secret sharing.

We use upper-case characters to denote functions. For a given unshared function $b = F(a)$, we generate a shared vector $\mathbf{F} = F_1, \ldots, F_s$ of component functions F_i in order to perform the shared calculation. The sharing $\mathbf{F}$ is *correct* if $b = \bigoplus_s b_i$ for $b_i = F_i(\mathbf{a})$. The algebraic degree of a function is denoted with t.

Adversarial model. We use d probing as our adversarial model which we define as follows. The adversary is allowed to probe d wires in the circuit within a certain time window. Each probed wire g calculating a function G gives information about all the inputs of G up to the latest synchronization point[1]. This definition

[1] One of the many ways of synchronization is storing elements in registers which we inherit throughout this paper without loss of generalization.

directly implies that the adversary can derive all the intermediate values during the computation of G and hence the output of G. To clarify, let us refer to Figure 1. An attacker probing the output of the function G (that is, wire g) can observe all the inputs to G up to, and including, reg_2; can generate all the intermediate values used during the calculation of G (including the outputs of G that are stored in reg_3); but can not directly learn all the values stored in reg_1 or any intermediate values occurring during the calculations of each F_i.

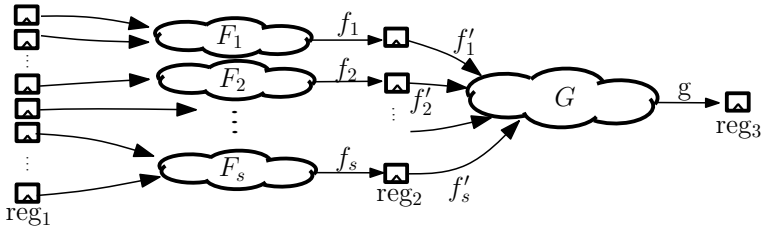


Figure 1 – Exemplary circuit to aid the explanation of the adversarial model adopted in this paper. F_i and G are combinational logic blocks, reg_i are register stages and f_i and g are wires that compute the F_i (resp. G) functions.

We note that this is a theoretical model stronger than a real-world attacker since a real-world attacker can only get a subset of the mentioned information. Moreover, it is slightly different than the conventional d-probing model [ISW03]. Nevertheless, it is advantageous since being able to see the inputs of the gates used during the calculation implies the ability to observe real world effects such as glitches. Hence, we gain the flexibility to work also with non-ideal (glitchy) gates. If the usage of ideal gates is assumed, working with the conventional model is typically sufficient.

This model matches quite nicely with d^{th}-order DPA attacks, which consider a noisy function of intermediates' leakage [DDF14]. There are certainly other adversarial models that are even more powerful, in which the attacker has the ability to adaptively move the probes between time periods (but not within a time period). We note that this "adaptive-probes" model is stronger and we do not consider moving probes in this paper.

Ishai–Sahai–Wagner scheme. Private circuits [ISW03] provide a procedure for computation on masked data. They give a construction for NOT and AND gates, and prove the security against d probes of any circuit composed of these secure gates ("gadgets") which are in turn built from logic gates that do not glitch. To compute $c = F(a, b) = ab$ while providing security against d probes, ISW takes $s = 2d + 1$ shares $\mathbf{a}$ and $\mathbf{b}$ of each input and consumes $\binom{s}{2}$ bits of randomness. We exemplify the computation of a masked AND gate providing security against adversaries using one ($d = 1, s = 3$) and two ($d = 2, s = 5$) probes in Equation (1),

Require: s-shares **a** and **b**
Ensure: s-shares **c** satisfying $c = ab$
  for i from 1 to s **do**
    for j from $i+1$ to s **do**
      $z_{ij} \leftarrow$ `rnd()`
      $z_{ji} \leftarrow (z_{ij} \oplus a_i b_j) \oplus a_j b_i$
    end for
  end for
  for i from 1 to s **do**
    $c_i \leftarrow a_i b_i$
    for j from 1 to s, $j \neq i$ **do**
      $c_i \leftarrow c_i \oplus z_{ij}$
    end for
  end for

Figure 2 – ISW algorithm.

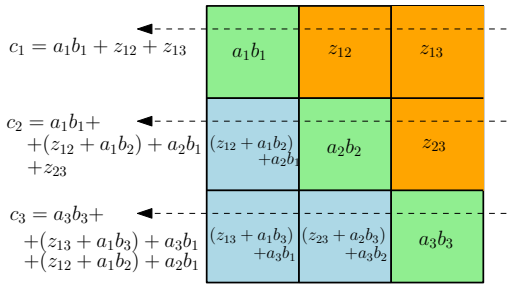


Figure 3 – Intermediate state of the ISW computation for $s = 3$.

and Equations (2) and (3) respectively. An intermediate state of the computation is shown in Figure 3.

$$\begin{aligned} z_{21} &= (z_{12} \oplus a_1b_2) \oplus a_2b_1, & c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13}, \\ z_{31} &= (z_{13} \oplus a_1b_3) \oplus a_3b_1, & c_2 &= a_2b_2 \oplus z_{21} \oplus z_{23}, \\ z_{32} &= (z_{23} \oplus a_2b_3) \oplus a_3b_2, & c_3 &= a_3b_3 \oplus z_{31} \oplus z_{32}. \end{aligned} \tag{1}$$

First, three (resp. ten) bits of randomness z_{ij} where $1 \leq i < j \leq s$ are drawn i.i.d. uniformly random. Then the intermediates z_{ji} are computed as shown in the left column of Equation (1) (resp. Eqn. (2)). The last step xors the intermediates z_{ij} and the products a_ib_i to compute the s output shares **c** (right column of Eqn. (1) and Eqn. (3) respectively).

$$\begin{aligned} z_{21} &= (z_{12} \oplus a_1b_2) \oplus a_2b_1, & z_{31} &= (z_{13} \oplus a_1b_3) \oplus a_3b_1, \\ z_{41} &= (z_{14} \oplus a_1b_4) \oplus a_4b_1, & z_{51} &= (z_{15} \oplus a_1b_5) \oplus a_5b_1, \\ z_{32} &= (z_{23} \oplus a_2b_3) \oplus a_3b_2, & z_{42} &= (z_{24} \oplus a_2b_4) \oplus a_4b_2, \\ z_{52} &= (z_{25} \oplus a_2b_5) \oplus a_5b_2, & z_{43} &= (z_{34} \oplus a_3b_4) \oplus a_4b_3, \\ z_{53} &= (z_{35} \oplus a_3b_5) \oplus a_5b_3, & z_{54} &= (z_{45} \oplus a_4b_5) \oplus a_5b_4. \end{aligned} \tag{2}$$

$$\begin{aligned} c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13} \oplus z_{14} \oplus z_{15}, & c_4 &= a_4b_4 \oplus z_{41} \oplus z_{42} \oplus z_{43} \oplus z_{45}, \\ c_2 &= a_2b_2 \oplus z_{21} \oplus z_{23} \oplus z_{24} \oplus z_{25}, & c_5 &= a_5b_5 \oplus z_{51} \oplus z_{52} \oplus z_{53} \oplus z_{54}. \\ c_3 &= a_3b_3 \oplus z_{31} \oplus z_{32} \oplus z_{34} \oplus z_{35}, \end{aligned} \tag{3}$$

Extensions to higher orders are similarly generated using the algorithm in Figure 2.

It is well known that the ISW algorithm can work in larger finite fields by building upon field multiplications instead of AND gates. In the case of AES, there is a significant performance gain if ISW operates in GF(2^8), due to the algebraic structure of the AES S-box [RP10]. We refer to [ISW03] for a variant of this method using $s = d + 1$ shares.

Threshold Implementations. TI provides provable security against d^{th}-order DPA even in a circuit with glitches according to [BGN+14b]. In addition, it is also advantageous since any degree t function can be securely implemented using at least $s \geq td + 1$ shares.

The security of a single function relies on the satisfaction of *d^{th}-order non-completeness*: any combination of up to d component functions F_i of $\mathbf{F}$ must be independent of at least one input share. It is shown that such a sharing can always be constructed using $s_{in} = td + 1$ and $s_{out} = \binom{s_{in}}{t}$ shares. Examples for the function $d = F(a, b, c) = c \oplus ab$ are given in Equations (4) and (5) for $d = 1$ and $d = 2$ respectively.

$$\begin{aligned} d_1 &= c_2 \oplus a_2b_2 \oplus a_1b_2 \oplus a_2b_1, \\ d_2 &= c_3 \oplus a_3b_3 \oplus a_3b_2 \oplus a_2b_3 \\ d_3 &= c_1 \oplus a_1b_1 \oplus a_1b_3 \oplus a_3b_1. \end{aligned} \tag{4}$$

Notice that $s_{out} > s_{in}$ for $d > 1$. In order to avoid further increase of the number of shares when several functions are cascaded, some of the output shares are typically xored. It is important that this reduction is performed only after the s_{out}-sharing $\mathbf{d}$ is stored in the registers in order to satisfy the non-completeness property and to avoid glitches depending on all shares of a variable.

$$\begin{aligned} d_1 &= c_2 \oplus a_2b_2 \oplus a_1b_2 \oplus a_2b_1, & d_2 &= c_3 \oplus a_3b_3 \oplus a_1b_3 \oplus a_3b_1, \\ d_3 &= c_4 \oplus a_4b_4 \oplus a_1b_4 \oplus a_4b_1, & d_4 &= c_1 \oplus a_1b_1 \oplus a_1b_5 \oplus a_5b_1, \\ d_5 &= a_2b_3 \oplus a_3b_2, & d_6 &= a_2b_4 \oplus a_4b_2, \\ d_7 &= c_5 \oplus a_5b_5 \oplus a_2b_5 \oplus a_5b_2, & d_8 &= a_3b_4 \oplus a_4b_3, \\ d_9 &= a_3b_5 \oplus a_5b_3, & d_{10} &= a_4b_5 \oplus a_5b_4. \end{aligned} \tag{5}$$

In order to provide security when several functions are cascaded, (i.e. the output of $\mathbf{F}$ is used as the input to another shared nonlinear-function $\mathbf{G}$), the shared function and its output should satisfy *uniformity* [BGN+14a]. Several methods to achieve uniformity have been proposed [BNN+12, BNN+15, KNP13, PMK+11]. It is advised to use re-masking [BGN+14a, MPL+11] in case these methods do not provide a solution.

When a single AND gate is considered, it has been shown that there exists no 3-sharing satisfying both uniformity and first-order non-completeness [BNN+12]. Therefore, the output shares of the 3-share AND gate must be re-masked (refreshed). Moreover, the sharing in Equation (4) considering an AND and XOR gate instead of a single AND gate satisfies all TI properties.

Trichina AND-gate. Unlike ISW and TI which can be applied both at the algorithm and at the gate level, Trichina [Tri03] investigates how to implement a masked AND gate $c = ab$ securely strictly at the gate level. The construction, which is described in Equation (6), requires two 2-share inputs and uses 1-bit extra randomness z_{12} to generate a 2-share output. The security relies strictly on the order of the operations to avoid unmasking certain bits, on the ideality of the cells and on the assumption that the sharing **a** of a is independent from **b**.

$$\begin{aligned} c_1 &= (((z_{12} \oplus a_1 b_2) \oplus a_2 b_1) \oplus a_2 b_2) \oplus a_1 b_1 \\ c_2 &= z_{12}. \end{aligned} \tag{6}$$

3 Conciliation

In this section we first relate the ISW scheme with TI and the Trichina gate using elementary transformations. We then use ingredients from all three schemes to describe a generalized masking construction and argue its security. As a case study, we consider a first-order sharing of an AND gate.

3.1 From ISW to TI

Consider the ISW construction with $s = 3$ input shares, providing first-order security as depicted in Figure 4. It is equivalent to the computation in Equation (1) and to Figure 2. In Figure 4, the computation flows from the outside towards the center. It begins with deriving all the cross products $a_i b_j$. Then three random values z_{ij} are added to some of the cross products. The terms are finally xored together in three groups to generate the output shares c_i.

In the following, we perform several elementary transformations on this circuit to arrive to a typical re-masked 3-share TI of an AND gate.

First transformation: moving random bits. Delaying the injection of randomness using the random bits z_{ij} closer to the center (towards the end of the calculation) as depicted in red yields the construction in Figure 5. Of course, this operation preserves the correctness of the output. It is already possible to recognize a

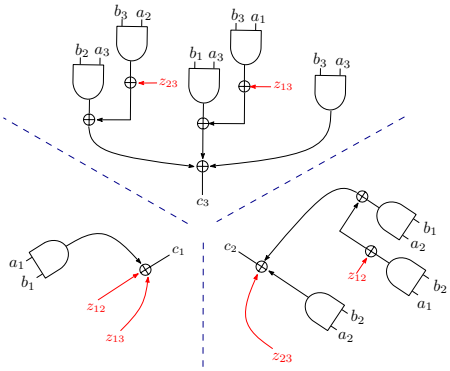


Figure 4 – Original ISW scheme.

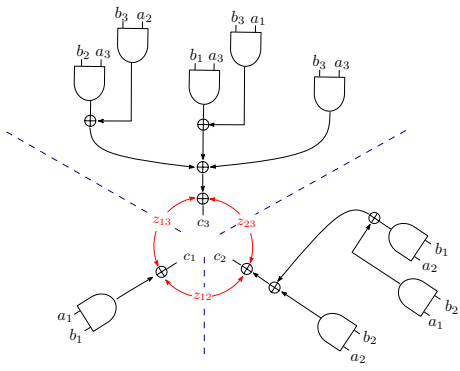


Figure 5 – After first transformation.

refreshing operation in the inner ring where z_{12}, z_{13} and z_{23} are involved. Note that the security of this intermediate construction highly depends on the order of computation of the XOR gates and the ideality (glitching behavior) of the gates.

Second transformation: moving AND gates. The next modification transforms the circuit of Figure 6 into Figure 7. It simply moves around the two red AND gates a_1b_3 and a_3b_1 together with the XOR gate from the upper to the lower-left branch.

This second transformation also preserves the correctness at the output. Notice that after this transformation each branch of the circuit sees at most two shares of each input. For example, the upper branch sees only a_2, a_3, b_2 and b_3. We can absorb the computation from each branch (3 ANDs and 2 XORs) into its respective component function F_i as shown in Equation (7).

$$\begin{aligned} F_1(a_1, a_2, b_1, b_2) &= a_2b_2 \oplus a_1b_2 \oplus a_2b_1, \\ F_2(a_2, a_3, b_2, b_3) &= a_3b_3 \oplus a_2b_3 \oplus a_3b_2, \\ F_3(a_1, a_3, b_1, b_3) &= a_1b_1 \oplus a_3b_1 \oplus a_1b_3. \end{aligned} \tag{7}$$

The reader will recognize that the resulting sharing $\mathbf{F}$ is a TI (satisfying first-order non-completeness) followed by a refreshing (resulting from the first transformation.) Note that one could also equivalently see this refreshing as an addition with the uniform shares of the constant value 0.

The security of this construction follows from the fact that it is a TI, followed by a refreshing. In particular, this construction is secure even in the presence of glitches. Therefore, we link the $s = 3$ ISW scheme to first-order TI.

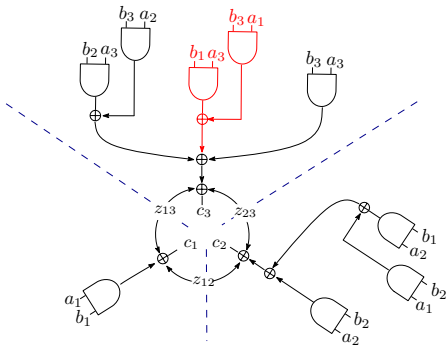


Figure 6 – Before second transformation.

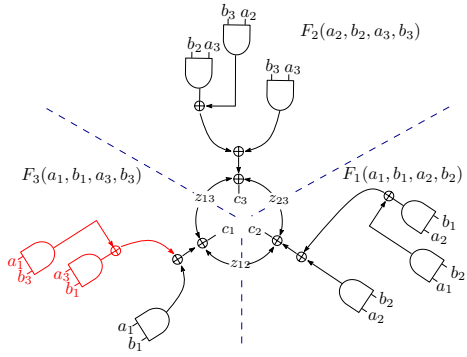


Figure 7 – After second transformation.

3.2 From ISW to the Trichina AND-gate

In Figure 8, we draw the ISW computation[2] of an AND gate for $s = 2$. In Figure 9, we have the Trichina AND gate. Similar to Section 3.1, we can transform the ISW construction $s = 2$ to the Trichina gate by rearranging the term $a_1 b_1$. It is noteworthy that the Trichina gate resembles a simplified ISW, and thus can be seen as the practitioners version.

3.3 Generalizing and inducing a structure

We can generalize the masked AND-gate transformations from Sections 3.1 and 3.2 to the general case of higher orders. In addition, we can construct variants that compute logic gates with more than two inputs or more sophisticated functions.

In order to induce a structure to the mentioned generalization, we decompose the resulting construction into four layers as exemplified in Figures 10 and 11 for first- and second-order security respectively. Specifically, we notice a non-linear layer $\mathcal{N}$, followed by a linear layer $\mathcal{L}$ and a refreshing layer $\mathcal{R}$. In certain cases where we want to reduce the number of shares, we add a linear compression layer $\mathcal{C}$. Below we detail the functionality of each layer.

The non-linear layer $\mathcal{N}$. This layer is responsible for the bulk of the computation, corresponding to the cross products $a_i b_j$. For example, in the first-order secure construction of a 3-share 2-input AND gate, $\mathcal{N}(\mathbf{a}, \mathbf{b}) = (a_1 b_1, a_1 b_2, \ldots, a_3 b_3)$ maps

[2] We stress that ISW with $s = 2$ is neither strictly defined nor proven secure in the ISW simulation model. We are extending the algorithm in Figure 2 in a straight forward way to any s.

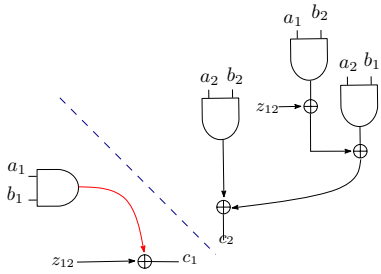


Figure 8 – ISW with $s = 2$.

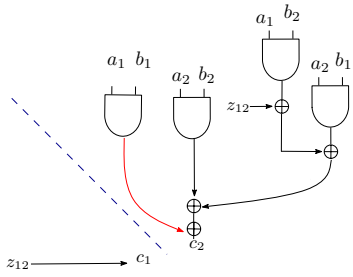


Figure 9 – Trichina AND gate.

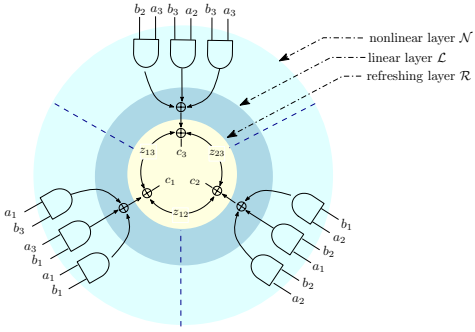


Figure 10 – First-order secure ($s = 3$) after transformation.

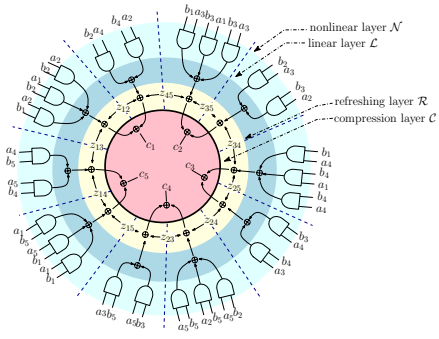


Figure 11 – Second-order secure ($s = 5$) after transformation.

6 input bits to 9 output bits ($a_i b_j$). Note that the set of cross products calculated in this layer is defined by the number of shares and the function itself. [3]

In order to generalize the construction such that a function other than $c = ab$ is computed (such as $d = abc$, $d = a \oplus bc$, $d = a \oplus bc \oplus abc$), $\mathcal{N}$ needs to be modified accordingly. To be specific, all the shared terms (cross products and linear terms) should be placed in $\mathcal{N}$ to be used in the following steps.

The linear layer $\mathcal{L}$. This layer is an XOR net that reduces the number of shares without modifying the unshared value. In the AND-gate example, it maps 9 input bits (output of $\mathcal{N}$) to 3 output bits for the first-order case and 25 input bits to 10 output bits in the second-order case. The linear layer $\mathcal{L}$ of TI is responsible for preserving non-completeness. Failure to achieve non-completeness can cause sensitive information leakage in a glitchy circuit as in the case of the original ISW scheme (see Section 4.2).

The reduction of the number of shares performed by $\mathcal{L}$ partially limits the exponential blow-up of shares otherwise caused by the non-linear layer $\mathcal{N}$ alone. We point out that the output of $\mathcal{N}$ is already a valid, non-complete sharing when each cross product is considered as an output share. However, cascading several sharings without $\mathcal{L}$ increases the number of shares rapidly, making such an implementation impractical except for circuits with very shallow logic depth.

The refreshing layer $\mathcal{R}$. This layer is applied in order to re-mask the output of $\mathcal{L}$. It is shown in several prior works on first-order TI that this layer can be avoided if the output of $\mathcal{L}$ already satisfies uniformity. However, $\mathcal{R}$ is critical in order to provide higher-order security as will be discussed in Section 4.1. In ISW, each output of each masked AND gate is refreshed, which clearly increases the randomness requirements compared to the generalized sharing of a more complex function with several terms (such as $d = a \oplus bc \oplus abc$).

The compression layer $\mathcal{C}$. While designing the $\mathcal{L}$ layer, there is a natural tension between two desirable properties, namely satisfying d^{th}-order non-completeness and having a small number of output shares. Normally, one designs $\mathcal{L}$ to have a small number of output shares yet satisfying d^{th}-order non-completeness. One example of this issue is the second-order masking of an AND-gate depicted in Figure 11, where the number of shares at the output of $\mathcal{L}$ (10 shares) is considerable larger than the number of shares of each input variable (5-share $\mathbf{a}$ and $\mathbf{b}$).

[3] It is possible to add other terms to $\mathcal{N}$ (as long as they are inserted an even number of times), such as virtual variable pairs [BNN+15], in order to increase the flexibility for generating a uniform sharing.

If it is desired to decrease the number of output shares further, the compression layer $\mathcal{C}$ is applied. This layer is composed of XOR gates only. In order to satisfy non-completeness and avoid glitches causing leakage of more than the intended number of shares, it is crucial to isolate the $\mathcal{R}$ and $\mathcal{C}$ layers using registers.

Note that in typical TIs, the layers $\mathcal{N}$ and $\mathcal{L}$ are combined and absorbed into the component functions without registers between these layers as drawn in Figure 11. An additional challenge is to design $\mathcal{L}$ so that it simultaneously satisfies non-completeness and uniformity.

3.4 Security arguments for generalized scheme

In this section, we argue the security of the generalized structure. We start by showing the security of a 2-input AND gate (2AND) against a d-probing adversary and inductively continue to a function of degree t. We assume that inputs to $\mathcal{N}$ are uniformly shared and synchronous. This discussion enables us to relate the number of required shares in TI with that in ISW.

2-input AND gate. Let us consider a set of information I (based on indices) gathered by the attacker by probing d wires. Specifically, if a wire corresponding to a_i, b_i or $a_i b_i$ is probed, the index $i \in I$. If the wire corresponding to $a_i b_j$ is probed, both $i, j \in I$. This implies that a probed wire at the output of the layer $\mathcal{N}$ can give information about at most two indices. Therefore, the cardinality of I is at most $2d$ when d wires are probed in $\mathcal{N}$. It follows that using at least $2d+1$ shares is required to provide security up to $\mathcal{L}$. However, an attacker is not limited to probing certain layers. Notice that the attacker probing closer to the end of the calculation of the component functions, i.e. just before the register between $\mathcal{R}$ and $\mathcal{C}$, gains more information. By the definition of the linear layer $\mathcal{L}$, the component functions should be formed such that any combination of up to d of them should be independent of at least one share, i.e. one index, when a d-probing secure circuit is considered. Hence, we know that if it is possible to construct $\mathcal{L}$ with the given restriction, the attacker probing d wires never has all the indexes in I. Since the input shares are uniformly shared and the vectors $\mathbf{a}_{\bar{\mathbf{i}}}, \mathbf{b}_{\bar{\mathbf{i}}}, \ldots$ are independent from the unshared values $a, b, \ldots$, we achieve security at the end of $\mathcal{L}$. Moreover, knowing the randomness used in $\mathcal{R}$ does not yield additional information to the attacker. At this point the possibility of generating $\mathcal{L}$ with $2d+1$ shares becomes the question. It has been shown in [BGN+14b] with a combinatorial argument that this is possible if the linear layer $\mathcal{L}$ is divided into $\binom{2d+1}{2}$ component functions. Namely, each component function uses at most two input shares and hence at most two indices are put in I for each probing. This gives the cardinality of at most $2d$ when d probes are used.

Notice that the security discussion provided so far considers only one AND gate. However, the security of the generalized scheme also holds for the composition of several AND gates. Namely, the refreshing layer $\mathcal{R}$ and the register afterwards impose independence of the composed elements and non-completeness respectively. Hence, the union of the gathered information does not give an additional advantage to the attacker.

In the case of a single-probe adversary, we can relax the requirements on $\mathcal{R}$. As long as the next nonlinear function sees uniformly shared inputs, one can simplify the construction of $\mathcal{R}$ and even in some cases avoid $\mathcal{R}$. This result follows from the fact that an attacker using a single probe is unable to combine information from more than one function.

3-input AND gate. The security argument for a 3-input AND gate ($F(a, b, c) = abc$) follows the same lines as for the 2-input AND gate. The nonlinear layer $\mathcal{N}$ calculates $a_i b_j c_k$ terms. In order to keep the number of shares small, we need to make sure that each component function uses variables with at most 3 different indices. Then, an attacker probing d wires can only gather information from at most $3d$ indices. The question if it is possible to arrange $\mathcal{L}$ such that this restriction is respected is answered positively in [BGN+14b]. It can be done by dividing $\mathcal{L}$ into $\binom{3d+1}{3}$ component functions. Note that for a full proof of security, the insertion of randomness (the $\mathcal{R}$ layer) and registers become critical in order to provide higher-order security of the composition of such gates.

Naturally, it is also possible to generate a shared 3AND gate by composing two shared 2AND gates. This requires usage of registers after both the first and the second 2AND-gate calculation. However, the construction described above which performs the 3AND gate calculation at once is typically more efficient.

t-input AND gate. We can inductively apply the arguments for 2AND and 3AND gates to the t-input AND gate. This implies the sufficient lower bound of $s_{in} = td+1$ input shares. The shared function should be split into at least $\binom{s_{in}}{t}$ component functions in $\mathcal{L}$ and satisfy d^{th}-order non-completeness.

Degree t Boolean functions. The above inductive argument does not exclude functions composed of more than one degree t term. To clarify, the generation of $\mathcal{N}$ is performed by straight-forward calculation of all cross products using $s_{in} = td + 1$ shares. The linear layer is split into $\binom{s_{in}}{t}$ component functions, each of which sees t indices as described above for a t-input AND gate. Any shared term of degree $\leq t$ can be placed to at least one existing component function since any cross product of the shared $\leq t$ term uses at most t indices, which concludes the argument.

Degree t functions in other finite fields. A careful investigation of the above arguments shows that they are independent of the used field. Namely, it is enough to replace the AND gates in GF(2) with multiplication in the required field in order to provide security for a degree t function.

We conclude the security argument of the generalized masking scheme by noting that s_{in} can be chosen to be greater than $td + 1$ in order to achieve flexibility without invalidating the security arguments.

3.5 Wrapping up.

In this section, we provided a generalized scheme which extends ISW and TI like masking schemes. Specifically, unlike the ISW scheme which builds up on AND gates or field multiplications; the generalized scheme allows to implement any function directly, enabling the usage of less compositions. The generalized scheme inherits the ability to operate on larger fields and security against d-probing adversary. In addition, it offers protection for composition of gates.

4 What can go wrong?

In Section 3 we constructed a generalized masking structure, and assigned precise requirements and functions to each of its layer. In this section, we show how small deviations from this generalized scheme can cause vulnerable implementations. In particular, in Section 4.1 we analyze the cost of lacking a refreshing layer $\mathcal{R}$. We use the recently proposed higher-order TI as our case study to show a higher-order flaw, then we suggest a generic fix. In Section 4.2, we elaborate on the insecurity that deviating from the structure especially on $\mathcal{L}$ brings in the presence of glitches using the ISW and Trichina scheme as our case study.

4.1 Higher-order TI is not so higher-order secure

The higher-order TI proposed in [BGN+14b] fits to our generalized structure as follows. $\mathcal{N}$ and $\mathcal{L}$ together enforce a correct and d^{th}-order non-complete implementation. However, unlike the generalized scheme, the refreshing layer $\mathcal{R}$ is not performed in TI when the uniformity of the shared output of $\mathcal{C}$ can be satisfied without $\mathcal{R}$. This difference becomes important since as we shall see in the sequel, it can induce a higher-order security flaw when composing several sharings, even if these sharings are uniform.

For simplicity, we use a second-order TI of a *mini-cipher* construction and show a second-order leakage.

Description of the mini-cipher. Let us consider a minimal non-linear feedback shift register. This mini-cipher comes from an extreme simplification of the KATAN block cipher for which a higher-order TI was given in [BGN+14b]. We consider a 4-bit state $S[i]$, $i \in 0, \ldots, 3$ for which the taps are at the state bits with indices $i = 1, 2, 3$ and the feedback is plugged at position 0. This state is a "sensitive variable[4]". The feedback function ("round function" of an extremely unbalanced Feistel) $F = F(S[3], S[2], S[1])$ is the same AND-XOR feedback function as in KATAN, namely $d = F(a, b, c) = ab \oplus c$.

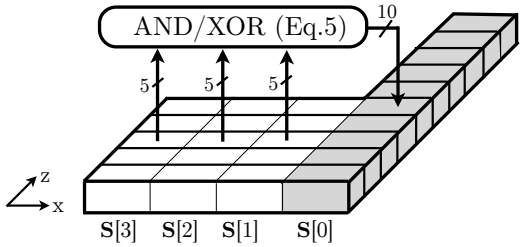


Figure 12 – Diagram of the shared version of the mini-cipher.

Shared version of the mini-cipher. The shared version of this mini-cipher (non-complete sharing in the $\mathcal{N}$ and $\mathcal{L}$) follows the lines of [BGN+14b]. The feedback function F is shared as Equation (5). In particular, to provide security against glitches, the state bit $\mathbf{S}[0]$, in which the output of $\mathbf{F}$ is stored, is composed of 10 shares, whereas $\mathbf{S}[1], \mathbf{S}[2], \mathbf{S}[3]$ are composed of 5 shares. The conversion from 10 to 5 shares is done as suggested in [BGN+14b]. That is, the fifth share of $\mathbf{S}[1]$ sees the xor of the last six shares of $\mathbf{S}[0]$ when the cipher is clocked.

A second-order leakage. Some lengthy, albeit straightforward, calculations show that the covariance (second mixed statistical moment) between the fifth share of $\mathbf{S}[1]$ after the first cycle and the fourth share of $\mathbf{S}[1]$ after the seventh cycle depends on the unshared initial value $S[2]$. Thus, there is a second-order flaw that invalidates the security claims of the scheme.[5]

Mitigation. The direct mitigation is to refresh the shares after each shared function computation, for example by adding fresh shares of the null vector. In other words, the refreshing layer $\mathcal{R}$ should be implemented in TI when higher-order security is considered. The idea here is to isolate the intermediates occurring within

[4]The goal is to show leakage in this construction. To simplify and keep the essentials, we do not explicitly inject the key in this mini-cipher construction, but assume that the initial state is secret (sensitive).

[5]This result had been previously reported in [Rep] and experimentally confirmed in [SM15].

each computation stage from intermediates of another stage, so that combining intermediates from different stages no longer reveals information about a secret unshared value. With this argument, we fix the flaw in [BGN+14b] using the conciliation of masking schemes.

Note that this fix naturally increases area and randomness requirements and we do not claim that it is the optimal solution. We foresee that this solution may be an overkill in many situations, and a careful analysis can save significant amount of randomness. This is especially true since the existence of a second-order dependency between two variables does not necessarily imply an easy key-extraction by DPA. In particular, if there is a second-order flaw between two intermediates for which there is enough key-diffusion between them, key recovery exploiting the joint leakage of intermediates becomes difficult. The exact minimum amount of $\mathcal{R}$ layers needed to make the whole implementation secure against higher-order attacks may depend from design to design.

4.2 ISW and Trichina in the presence of glitches

ISW scheme implicitly considers a logic gate that does not glitch. Thus, a straightforward translation of ISW into standard CMOS technology can result in a vulnerable implementation. To see this, observe that in Equation (8) c_3 breaks the non-completeness property:

$$
\begin{aligned}
c_1 &= a_1b_1 \oplus z_{12} \oplus z_{13},\\
c_2 &= a_2b_2 \oplus ((z_{12} \oplus a_1b_2) \oplus a_2b_1)) \oplus z_{23},\\
c_3 &= a_3b_3 \oplus ((z_{13} \oplus a_1b_3) \oplus a_3b_1) \oplus (z_{23} \oplus a_2b_3) \oplus a_3b_2.
\end{aligned}
$$

The trivial fix here is to register signals that otherwise could result in undesired (and pernicious) glitches. More precisely, if during the ISW computation in logic, the intermediate values z_{ji} where $i < j$ (outputs in Equation (1), left column; and Equation (2)) are stored in registers together with the intermediate values a_ib_i before further XOR combinations, the circuit becomes secure even if there are glitches. This follows since non-completeness holds between register stages. The caveat of this fix is the significant increase in area (and latency) due to extra registers. Note that this extra layer of registers is prevented by careful selection of the layer $\mathcal{L}$.

Similar observations apply to the Trichina construction. Trichina also imposes restrictions on the logic gates, especially on the order of evaluation of these gates. It is implicitly assumed that signals are registered or latched in order to avoid glitches. The case where first-order security fails due to glitches is studied in [MS06].

5 Applications

Here we introduce two additional constructive applications. In the first one, we focus on optimizing the generalized scheme further such that it uses less input shares per function. The second application analyses software-like implementations in a distance-based leakage model.

5.1 Using $d+1$ input shares

As described in Section 3, the generalized scheme uses at least $td+1$ input shares to protect a function with degree t against d-probing attacks. Here, we improve the scheme such that it uses less input shares, specifically, $d+1$ shares to achieve d-probing security. As a trade-off, however, this sharings are more restrictive with the requirements of *independent* input sharings. We illustrate with single-probe secure examples the design process of such sharings and the construction of layers. We provide a security argument and discuss connections with prior works.

First crack. We start with the first-order sharing of $c = ab$ with $s_{in} = 2$ and $s_{out} = 4$ given in Equation (8). The sharing $\mathbf{c}$ is only composed of the crossproducts $a_i b_j$. Hence, it can be seen as the output of $\mathcal{N}$ which is already a correct sharing for c. Moreover, if the sharing of a is independent than that of b then each share c_i is independent of at least one input share of each variable. In other words, non-completeness is satisfied. This implies the independence of $\mathbf{c}$ from the unmasked variables a and b providing security under a single probe.

$$c_1 = a_1 b_1, \qquad c_2 = a_1 b_2, \qquad c_3 = a_2 b_1, \qquad c_4 = a_2 b_2. \tag{8}$$

Note that in this simple sharing, if the sharings of a and b were dependent (for example, $\mathbf{a} = \mathbf{b}$), then the second output share $a_1 b_2 = a_1 a_2$ would depend on all shares of a (breaking non-completeness) and this would clearly leak information about a. During the construction of layers in the following, we assume that the sharings of each input variable is independent from all others.

Construction of $\mathcal{N}$ and $\mathcal{L}$. The increase of number of variables in the input increases the number of cross products and hence the number of output shares of $\mathcal{N}$ exponentially. For example, if we consider the sharing of $d = a \oplus ac \oplus bc$ with $s_{in} = 2$, we end up with 10 terms $(a_1, a_2, a_1c_1, a_1c_2, a_2c_1, a_2c_2, b_1c_1, b_1c_2, b_2c_1, b_2c_2)$ in $\mathcal{N}$. Notice that it is possible to reduce the number of output shares using a careful selection of a linear layer $\mathcal{L}$ as shown in Equation (9) while satisfying the non-completeness property.

$$\begin{aligned} d_1 &= a_1 \oplus a_1c_1 \oplus b_1c_1, & d_3 &= a_2 \oplus a_2c_1 \oplus b_2c_1, \\ d_2 &= a_1c_2 \oplus b_1c_2, & d_4 &= a_2c_2 \oplus b_2c_2. \end{aligned} \tag{9}$$

The number of output shares of $\mathcal{L}$ also changes significantly depending on the function itself in addition to the number of input shares. To clarify, let us consider the sharing of $d = a \oplus ac \oplus bc \oplus ab$ which differs from the prior unshared function in the additional term ab. The terms $(a_1b_1, a_1b_2, a_2b_1, a_2b_2)$ should be added to Equation (9) for a correct implementation. Even if we place the additional terms a_1b_1 and a_2b_2 to the first and the last component functions in Equation (9) respectively, the remainding terms a_1b_2 and a_2b_1 can not be placed in these four component functions without breaking the non-completeness property. Hence, we need to increase the number of shares. One option to obtain non-completeness is increasing the number of output shares of $\mathcal{L}$ as shown in the equation below.

$$\begin{aligned} d_1 &= a_1 \oplus a_1c_1 \oplus b_1c_1 \oplus a_1b_1, & d_4 &= a_2c_2 \oplus b_2c_2 \oplus a_2b_2, \\ d_2 &= a_1c_2 \oplus b_1c_2, & d_5 &= a_1b_2, \\ d_3 &= a_2 \oplus a_2c_1 \oplus b_2c_1, & d_6 &= a_2b_1. \end{aligned} \tag{10}$$

Construction of $\mathcal{R}$ and $\mathcal{C}$. It is clear that if $n \times s_{in} < m \times s_{out}$, the output sharing can not be uniform. Even if $n \times s_{in} \geq m \times s_{out}$ the uniformity is not guaranteed. The output sharing described in Equations (8), (9) and (10) are such non-uniform examples which require refreshing ($\mathcal{R}$ layer). An alternative approach for the first-order case only is to decrease the number of shares in order to achieve uniformity after storing the output of the mentioned sharings in registers (prior to the $\mathcal{C}$ layer). To exemplify, consider the following sharing of $d = ab \oplus c$ with 2 input shares of each variable.

$$d_1 = a_1b_1 \oplus c_1, \qquad d_2 = a_1b_2, \qquad d_3 = a_2b_1 \oplus c_2, \qquad d_4 = a_2b_2. \tag{11}$$

The sharing $\mathbf{d}$ is a nonuniform 4-sharing. However, the 2-sharing $\mathbf{e}$ generated by $e_1 = d_1 \oplus d_2$ and $e_2 = d_3 \oplus d_4$ is uniform. Moreover, if the sharing $\mathbf{d}$ is stored in registers before decreasing the number of shares, as implied by the registers between the $\mathcal{R}$ and $\mathcal{C}$ layers in the generalized construction, any glitch during the calculation of e_i does not reveal information about the input values. Note that the selection of the xored terms is not random at all and must be performed with extreme care. A bad choice for a compression layer would be $e_1 = d_1 \oplus d_3$ and $e_2 = d_2 \oplus d_4$, since $e_2 = (a_1 \oplus a_2)b_2 = ab_2$ obviously reveals information on a.

Security argument of the improved bound on the number of shares. It is noteworthy that the security of the improved scheme is not proven using indices as for the generalized scheme described in Section 3.4. Instead, since we assume that each input sharing is independent of the others, we ensure that any combination of

d probes miss at least one share of each input variable. Therefore $d+1$ input shares are sufficient in order to provide non-completeness in $\mathcal{N}$ hence independence of the output shares from each unmasked input. As discussed above the requirements that should be satisfied by the $\mathcal{L}$ and $\mathcal{C}$ layers in order to provide the claimed security highly depends on the function. Therefore, we avoid to give a generic construction besides imposing d^{th}-order non-completeness in $\mathcal{L}$ and extreme care not to unmask in $\mathcal{C}$. The extension to higher orders is straightforward under given exceptions.

Application to 4-bit quadratic permutations. In order to increase the usability of this technique, we provide a possible sharing $\mathbf{S}$ secure against 2-probing adversary for one permutation S from each 4-bit quadratic affine equivalence class of permutation in Section 7. Any other permutation S' that is affine equivalent to S can be calculated by $S' = A \circ S \circ B$ where A and B refer to affine transformations which can be shared as described in Section 2. Note that there exist 6 such quadratic classes which can be used to generate half of the 4-bit cubic permutations as described in [BNN+15]. This set of permutations covers a significant part of all 4×4 S-boxes.

Connections with software ISW. In [RP10], a fast masked AES at any order is given. The authors improve the security guarantees with respect to the number of shares from $s = 2d + 1$ to $s = d + 1$. This improvement actually resembles to the contribution of this section. The improvement was later shown to be slightly flawed by [CPRR13]. However, we observe here that the refreshing from [RP10] is not exactly the same as the layer $\mathcal{R}$ presented in Section 3.3 (operating in $\mathrm{GF}(2^8)$). Namely, the refreshing from [RP10] uses 1 unit of randomness (elements in $\mathrm{GF}(2^8)$) less than $\mathcal{R}$. Using a refreshing that mimics the layer $\mathcal{R}$ makes the second-order flaw disappear [BBD+15].

5.2 Resistance against distance leakage

There are many applications of the ISW scheme for masked software implementations [RP10], [GLSV14]. In the case of AES, there is a significant performance gain if the ISW operates in $\mathrm{GF}(2^8)$, due to the algebraic structure of the AES Sbox. A common problem with ISW-based software implementations is the mismatch between the probing model in which ISW is proven secure and the leakage behavior of the device that runs the implementation. For instance, typical processors can be approximately modeled by a distance-based leakage behavior (Hamming distance) rather than value-based one (Hamming weight). Thus, a straightforward implementation of ISW without a careful prior profiling of the device leakage behavior will likely lead to an insecure implementation. This is because, even if ISW is secure in weight-based leakage behavior, it is not in a distance-based one.

There are already some theoretical solutions for this problem, although they come with a significant cost [CGP+12], [BGG+14]. We point out here that it is possible to minimize the exposure to this issue with the same modification performed in Section 3, i.e. bringing the non-completeness condition.

The generalized scheme (e.g. after the second modification in Figure 7) computes sequentially each branch (component function) $F_i, i = 1, 2, 3$ and then performs a refreshing. This scheme is secure even if during the computation of each branch F_i the device leaks distances (or a more complex leakage function of several values). Contrary to the ISW, we do not impose specific constraints on the order of evaluation of intermediates (within each F_i). This result immediately follows from non-completeness of each branch F_i. It is required, however, to make sure that there is no distance leakage between an intermediate appearing in F_i and another in F_j, for $i \neq j$. It is noteworthy that the randomness requirement, running time and memory requirements stay the same as in the original algorithm.

6 Conclusion

In this paper, we explored the connections, similitudes and differences between several masking schemes, both from theoretical domains and from practitioners working under real-world constraints. It is remarkable how two substantially disparate communities arrive to essentially similar designs. This perhaps builds even more confidence on the underlying techniques.

There are certainly many future avenues of research. For example, it would be desirable to have explicit and tight bounds on the randomness requirements to achieve efficient masked implementations.

Acknowledgements. The authors would like to thank the CRYPTO 2015 reviewers for their valuable comments, as well as Fré Vercauteren and Vincent Rijmen for stimulating discussions. This work has been supported in part by the Research Council of KU Leuven (OT/13/071 and GOA/11/007), by the FWO G.0550.12N and by the Hercules foundation (AKUL/11/19). Oscar Reparaz is funded by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO). Benedikt Gierlichs is a Postdoctoral Fellow of the Fund for Scientific Research - Flanders (FWO). Begül Bilgin is partially supported by the FWO project G0B4213N.

References

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified Proofs of Higher-

Order Masking. In *EUROCRYPT*, volume 9056 of *LNCS*, pages 457–485. Springer, 2015.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BGN+14a] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A More Efficient AES Threshold Implementation. In *AFRICACRYPT*, volume 8469 of *LNCS*, pages 267–284. Springer, 2014.

[BGN+14b] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BNN+12] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, and Georg Stütz. Threshold implementations of all 3×3 and 4×4 s-boxes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 76–91. Springer, 2012.

[BNN+15] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia Tokareva, and Valeriya Vitkup. Threshold implementations of small S-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.

[CGP+12] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, Soline Renner, Matthieu Rivain, and Praveen Kumar Vadnala. Conversion of Security Proofs from One Leakage Model to Another: A New Issue. In *COSADE*, volume 7275 of *LNCS*, pages 69–81. Springer, 2012.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara,*

California, USA, August 15-19, 1999, Proceedings, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[Cor14] Jean-Sébastien Coron. Higher Order Masking of Look-Up Tables. In Phong Q Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-Order Side Channel Security and Mask Refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[DDF14] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying Leakage Models: From Probing Attacks to Noisy Leakage. In *EUROCRYPT*, volume 8441 of *LNCS*, pages 423–440. Springer, 2014.

[GLSV14] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Var\ic\i. LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations. In *Fast Software Encryption - FSE 2014*, Londres, United Kingdom, mar 2014.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KNP13] Sebastian Kutzner, Phuong Ha Nguyen, and Axel Poschmann. Enabling 3-Share Threshold Implementations for all 4-Bit S-Boxes. In *ICISC*, pages 91–108. Springer, 2013.

[Mes00] Thomas S. Messerges. Using second-order power analysis to attack DPA resistant software. In Ç K Koç and C Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *LNCS*, pages 238–251. Springer, 2000.

[MPG05] Stefan Mangard, Thomas Popp, and Berndt M Gammel. Side-Channel Leakage of Masked CMOS Gates. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[MPL+11] A Moradi, A Poschmann, S Ling, C Paar, and H Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *EUROCRYPT*, volume 6632 of *LNCS*, pages 69–88. Springer, 2011.

[MS06] S Mangard and K Schramm. Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations. In *CHES*, volume 4249 of *LNCS*, pages 76–90. Springer, 2006.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold Implementations Against Side-Channel Attacks and Glitches. In Peng Ning, Sihan Qing, and Ninghui Li, editors, *Information and Communications Security, 8th International Conference, ICICS 2006, Raleigh, NC, USA, December 4-7, 2006, Proceedings*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[PMK+11] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-Channel Resistant Crypto for Less than 2,300 GE. *Journal of Cryptology*, 24(2):322–345, 2011.

[Rep] Oscar Reparaz. A note on the security of Higher-Order Threshold Implementations. Cryptology ePrint Archive, Report 2015/001.

[RGV12] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 155–174. Springer, 2012.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert,

editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[SM15] Tobias Schneider and Amir Moradi. Leakage Assessment Methodology - a clear roadmap for side-channel evaluations. In *CHES*, LNCS. 2015.

[TAV02] Kris Tiri, Moonmoon Akmal, and Ingrid Verbauwhede. A dyamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards. In *ESSCIRC*, pages 403–406, 2002.

[Tri03] Elena Trichina. Combinational Logic Design for AES SubByte Transformation on Masked Data. *IACR Cryptology ePrint Archive*, 2003:236, 2003.

[TV04] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *DATE*, 2004.

7 First-order Masking of Quadratic 4-bit Permutations with $d+1$ Shares

We use the notation in [BNN+15] in order to represent a class. Namely, Q_i^j corresponds to a quadratic class of j bits with the class number i where the classes are order lexicographically from their representatives. Each permutation $S(a,b,c,d) = (x,y,z,t)$ has 4 input and output bits. The component functions F, G, H, K outputs x, y, z, t respectively. x (resp. a) is the most significant bit whereas t (resp. d) is the least significant bit. We only consider first-order security with 2 input shares. The sharing $\mathbf{x}$ of an output variable x refers to its sharing after N followed by L_1. $\mathbf{x}$ is in some cases not uniform and requires refreshing if used as is. We also describe the 2-sharing $\bar{\mathbf{x}}$ with shares $\bar{x}_i$ after L_2 layer such that the sharing is uniform.

7.1 Class Q_4^4

$S = [0,1,2,3,4,5,6,7,8,9,10,11,13,12,15,14]$

$$
\begin{aligned}
x &= F(a,b,c,d) = a \\
y &= G(a,b,c,d) = b \\
z &= H(a,b,c,d) = c \\
t &= K(a,b,c,d) = ab \oplus d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 \\
x_2 &= F_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 \\
y_1 &= G_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_1 \\
y_2 &= G_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2 \\
z_1 &= H_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = c_1 \\
z_2 &= H_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = c_2 \\
t_1 &= K_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 b_1 \oplus d_1 \\
t_2 &= K_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 b_2 \\
t_3 &= K_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 b_1 \\
t_4 &= K_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 b_2 \oplus d_2
\end{aligned}
\qquad
\begin{aligned}
\bar{x}_1 &= x_1 \\
\bar{x}_2 &= x_2 \\
\bar{y}_1 &= y_1 \\
\bar{y}_2 &= y_2 \\
\bar{z}_1 &= z_1 \\
\bar{z}_2 &= z_2 \\
\bar{t}_1 &= t_1 \oplus t_2 \\
\bar{t}_2 &= t_3 \oplus t_4
\end{aligned}
$$

7.2 Class Q_{12}^4

$S = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 10, 11]$

$$
\begin{aligned}
x &= F(a, b, c, d) = a \\
y &= G(a, b, c, d) = ac \oplus b \\
z &= H(a, b, c, d) = ab \oplus ac \oplus c \\
t &= K(a, b, c, d) = d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a,b,c,d}) = a_1 \\
x_2 &= F_2(\mathbf{a,b,c,d}) = a_2 \\
y_1 &= G_1(\mathbf{a,b,c,d}) = a_1c_1 \oplus b_1 & \bar{x}_1 &= x_1 \\
y_2 &= G_2(\mathbf{a,b,c,d}) = a_1c_2 & \bar{x}_2 &= x_2 \\
y_3 &= G_3(\mathbf{a,b,c,d}) = a_2c_1 \oplus b_2 & \bar{y}_1 &= y_1 \oplus y_2 \\
y_4 &= G_4(\mathbf{a,b,c,d}) = a_2c_2 & \bar{y}_2 &= y_3 \oplus y_4 \\
z_1 &= H_1(\mathbf{a,b,c,d}) = a_1b_1 \oplus a_1c_1 \oplus c_1 & \bar{z}_1 &= z_1 \oplus z_2 \\
z_2 &= H_2(\mathbf{a,b,c,d}) = a_1b_2 \oplus a_1c_2 & \bar{z}_2 &= z_3 \oplus z_4 \\
z_3 &= H_3(\mathbf{a,b,c,d}) = a_2b_1 \oplus a_2c_1 & \bar{t}_1 &= t_1 \\
z_4 &= H_4(\mathbf{a,b,c,d}) = a_2b_2 \oplus a_2c_2 \oplus c_2 & \bar{t}_2 &= t_2 \\
t_1 &= K_1(\mathbf{a,b,c,d}) = d_1 \\
t_2 &= K_2(\mathbf{a,b,c,d}) = d_2
\end{aligned}
$$

7.3 Class Q_{293}^4

$S = [0, 1, 2, 3, 4, 5, 7, 6, 8, 9, 12, 13, 14, 15, 11, 10]$

$$
\begin{aligned}
x &= F(a, b, c, d) = a \\
y &= G(a, b, c, d) = ac \oplus b \\
z &= H(a, b, c, d) = ab \oplus ac \oplus c \\
t &= K(a, b, c, d) = bc \oplus d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 \\
x_2 &= F_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 \\
y_1 &= G_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_1 \oplus b_1 \\
y_2 &= G_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_2 \\
y_3 &= G_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_1 \oplus b_2 \\
y_4 &= G_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_2 \\
z_1 &= H_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus a_1c_1 \oplus c_1 \\
z_2 &= H_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 \oplus a_1c_2 \\
z_3 &= H_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 \oplus a_2c_1 \\
z_4 &= H_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus a_2c_2 \oplus c_2 \\
t_1 &= K_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_1c_1 \oplus d_1 \\
t_2 &= K_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_1c_2 \\
t_3 &= K_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_1 \oplus d_2 \\
t_4 &= K_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_2
\end{aligned}
\qquad
\begin{aligned}
\bar{x}_1 &= x_1 \\
\bar{x}_2 &= x_2 \\
\bar{y}_1 &= y_1 \oplus y_2 \\
\bar{y}_2 &= y_3 \oplus y_4 \\
\bar{z}_1 &= z_1 \oplus z_2 \\
\bar{z}_2 &= z_3 \oplus z_4 \\
\bar{t}_1 &= t_1 \oplus t_2 \\
\bar{t}_2 &= t_3 \oplus t_4
\end{aligned}
$$

7.4 Class Q^4_{294}

$S = [0, 1, 2, 3, 4, 5, 7, 6, 8, 9, 12, 13, 14, 15, 11, 10]$

$$
\begin{aligned}
x &= F(a,b,c,d) = a \\
y &= G(a,b,c,d) = b \\
z &= H(a,b,c,d) = ab \oplus c \\
t &= K(a,b,c,d) = ac \oplus d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a,b,c,d}) = a_1 \\
x_2 &= F_2(\mathbf{a,b,c,d}) = a_2 \\
y_1 &= G_1(\mathbf{a,b,c,d}) = b_1 \\
y_2 &= G_2(\mathbf{a,b,c,d}) = b_2 \\
z_1 &= H_1(\mathbf{a,b,c,d}) = a_1 b_1 \oplus c_1 \\
z_2 &= H_2(\mathbf{a,b,c,d}) = a_1 b_2 \\
z_3 &= H_3(\mathbf{a,b,c,d}) = a_2 b_1 \oplus c_2 \\
z_4 &= H_4(\mathbf{a,b,c,d}) = a_2 b_2 \\
t_1 &= K_1(\mathbf{a,b,c,d}) = a_1 c_1 \oplus d_1 \\
t_2 &= K_2(\mathbf{a,b,c,d}) = a_1 c_2 \\
t_3 &= K_3(\mathbf{a,b,c,d}) = a_2 c_1 \oplus d_2 \\
t_4 &= K_4(\mathbf{a,b,c,d}) = a_2 c_2
\end{aligned}
\qquad
\begin{aligned}
\bar{x}_1 &= x_1 \\
\bar{x}_2 &= x_2 \\
\bar{y}_1 &= y_1 \\
\bar{y}_2 &= y_2 \\
\bar{z}_1 &= z_1 \oplus z_2 \\
\bar{z}_2 &= z_3 \oplus z_4 \\
\bar{t}_1 &= t_1 \oplus t_2 \\
\bar{t}_2 &= t_3 \oplus t_4
\end{aligned}
$$

7.5 Class Q^4_{299}

$S = [0, 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 11, 9, 15, 13]$

$$
\begin{aligned}
x &= F(a, b, c, d) = a \\
y &= G(a, b, c, d) = ab \oplus ac \oplus b \\
z &= H(a, b, c, d) = ab \oplus ac \oplus ad \oplus c \\
t &= K(a, b, c, d) = ab \oplus ad \oplus d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 \\
x_2 &= F_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 \\
y_1 &= G_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus a_1c_1 \oplus b_1 \\
y_2 &= G_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 \oplus a_1c_2 \qquad & \bar{x}_1 &= x_1 \\
y_3 &= G_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 \oplus a_2c_1 \qquad & \bar{x}_2 &= x_2 \\
y_4 &= G_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus a_2c_2 \oplus b_2 \qquad & \bar{y}_1 &= y_1 \oplus y_2 \\
z_1 &= H_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus a_1c_1 \oplus a_1d_1 \oplus c_1 \qquad & \bar{y}_2 &= y_3 \oplus y_4 \\
z_2 &= H_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 \oplus a_1c_2 \oplus a_1d_2 \qquad & \bar{z}_1 &= z_1 \oplus z_2 \\
z_3 &= H_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 \oplus a_2c_1 \oplus a_2d_1 \qquad & \bar{z}_2 &= z_3 \oplus z_4 \\
z_4 &= H_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus a_2c_2 \oplus a_2c_2 \oplus c_2 \qquad & \bar{t}_1 &= t_1 \oplus t_2 \\
t_1 &= K_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus a_1d_1 \oplus d_1 \qquad & \bar{t}_2 &= t_3 \oplus t_4 \\
t_2 &= K_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 \oplus a_1d_2 \\
t_3 &= K_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 \oplus a_2d_1 \\
t_4 &= K_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus a_2d_2 \oplus d_2
\end{aligned}
$$

7.6 Class Q^4_{300}

$S = [0, 1, 2, 3, 4, 5, 8, 9, 13, 12, 7, 6, 11, 10, 15, 14]$

$$
\begin{aligned}
x &= F(a, b, c, d) = ac \oplus bc \oplus a \\
y &= G(a, b, c, d) = bc \oplus a \oplus b \\
z &= H(a, b, c, d) = ab \oplus bc \oplus c \\
t &= K(a, b, c, d) = a \oplus d
\end{aligned}
$$

$$
\begin{aligned}
x_1 &= F_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_1 \oplus b_1c_1 \oplus a_1 \\
x_2 &= F_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1c_2 \oplus b_1c_2 \\
x_3 &= F_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_1 \oplus b_2c_1 \oplus a_2 \\
x_4 &= F_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2c_2 \oplus b_2c_2 \\
y_1 &= G_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_1c_1 \oplus a_1 \oplus b_1 \qquad & \bar{x}_1 &= x_1 \oplus x_2 \\
y_2 &= G_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_1c_2 & \bar{x}_2 &= x_3 \oplus x_4 \\
y_3 &= G_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_1 & \bar{y}_1 &= y_1 \oplus y_2 \\
y_4 &= G_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_2 \oplus a_2 \oplus b_2 & \bar{y}_2 &= y_3 \oplus y_4 \\
z_1 &= H_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_1 \oplus b_1c_1 \oplus c_1 & \bar{z}_1 &= z_1 \oplus z_2 \oplus z_3 \\
z_2 &= H_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1b_2 & \bar{z}_2 &= z_4 \oplus z_5 \oplus z_6 \\
z_3 &= H_3(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_1 & \bar{t}_1 &= t_1 \\
z_4 &= H_4(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_1 & \bar{t}_2 &= t_2 \\
z_5 &= H_5(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = b_2c_1 \\
z_6 &= H_6(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2b_2 \oplus b_2c_2 \oplus c_2 \\
t_1 &= K_1(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_1 \oplus d_1 \\
t_2 &= K_2(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d}) = a_2 \oplus d_2
\end{aligned}
$$

Publication

Detecting flawed masking schemes with leakage detection tests

Publication Data

Oscar Reparaz. Detecting flawed masking schemes with leakage detection tests. In Thomas Peyrin, editor, *Fast Software Encryption, 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016*, volume 0000 of *Lecture Notes in Computer Science*, page 20. Springer, 2016.

Detecting flawed masking schemes with leakage detection tests

Oscar Reparaz

KU Leuven Dept. Electrical Engineering-ESAT/COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

Abstract. Masking is a popular countermeasure to thwart side-channel attacks on embedded systems. Many proposed masking schemes, even carrying "security proofs", are eventually broken because they are flawed by design. The security validation process is nowadays a lengthy, tedious and manual process. In this paper, we report on a method to verify the soundness of a masking scheme before implementing it on a device. We show that by instrumenting a high-level implementation of the masking scheme and by applying leakage detection techniques, a system designer can quickly assess at design time whether the masking scheme is flawed or not, and to what extent. Our method requires not more than working high-level source code and is based on simulation. Thus, our method can be used already in the very early stages of design. We validate our approach by spotting in an automated fashion first-, second- and third-order flaws in recently published state-of-the-art schemes in a matter of seconds with limited computational resources. We also present a new second-order flaw on a table recomputation scheme, and show that the approach is useful when designing a hardware masked implementation.

1 Introduction

Since Kocher published the seminal paper on side-channel attacks [Koc96], cryptographic embedded systems have been broken using some auxiliary timing information [Koc96], the instantaneous power consumption of the device [KJJ99] or the EM radiation [AARR02], among others. An attack technique of particular interest, due to its inherent simplicity, robustness and efficiency to recover secrets (such as cryptographic keys or passwords) on embedded devices is Differential Power Analysis (DPA), introduced in [KJJ99]. DPA relies on the fact that the instantaneous power consumption of a device running a cryptographic implementation is somehow dependent on the intermediate values occurring during the execution of the implementation. An especially popular countermeasure to

thwart power analysis attacks, including DPA, is masking [CJRR99,GP99]. Masking works by splitting every sensitive variable appearing during the computation of a cryptographic primitive into several shares, so that any proper subset of shares is independent of any sensitive variable. This, in turn, implies that the instantaneous power consumption of the device is independent of any sensitive variable, and thus vanilla DPA cannot be mounted. In theory, a $(d+1)$-order DPA attack can still be mounted against a d-th order masked implementation; however, in practice higher order DPA attacks are exponentially more difficult to carry out [CJRR99].

Crucially, in many cases the attacker is not required to perform a higher order attack because the masking is imperfect and thus does not provide the claimed security guarantees. The causes of the imperfections can be manifold: from implementation mistakes to more fundamental flaws stemming from the masking scheme itself. There are many examples in the literature of such flawed schemes: a "provably secure" scheme published in 2006 [PGA06] based on FFT and broken two years later [CGPR08], a scheme published in 2006 [SP06] and broken one year later [CPR07], another "provably secure" scheme published in 2010 [RP10] and (academically) broken three years later [CPRR13]; a scheme published in 2012 [BFGV12] and broken in 2014 [PRR14].

The verification process of a masking scheme is nowadays a very lengthy manual task, and the findings are published in solid papers involving convoluted probability arguments at leading venues, some years later after the scheme is published. Some even won a best paper award as [CPR07]. From the stand point of a system designer, it is often not acceptable to wait for a public scrutiny of the scheme or invest resources in a lengthy, expensive, evaluation.

Our contribution. In this paper we provide an automated method to test whether the masking scheme is sound or not, and to what extent. The method is conceptually very simple, yet powerful and practically relevant. We give experimental evidence that the technique works by reproducing state-of-the-art first-, second- and third-order flaws of masking schemes with very limited computational resources. Our approach is fundamentally different from previously proposed methodologies and is based on sampling and leakage detection techniques.

2 Leakage detection for masked schemes in simulation

Core idea. In a nutshell, our approach to detect flawed masking schemes is to simulate power consumption traces from a high-level implementation of the masking scheme and then perform leakage detection tests on the simulated traces to verify the first- and higher-order security claims of the masking scheme.

Input and output of the tool. The practitioner only ought to provide working source code of the masked implementation. The tool instruments the code, performs leakage detection tests and outputs whether the scheme meets its security goals or not. In addition, should a problem be detected, the tool pinpoints the variables causing the flaw and quantifies the magnitude of the statistical bias.

Security claims of masking schemes. We use in this paper the conventional notions for expressing the security claim of a masking scheme. Namely, a masking scheme provides first-order security if the expected value of each single intermediate does not depend on the key. More generally, a masking scheme provides k-order security if the k-th order statistical moment of any combination of k intermediates does not depend on the key. This formulation is convenient since leakage detection tests are designed specifically to test these claims.

Three steps. Our tool has three main ingredients: trace generation, trace preprocessing and leakage detection. We describe each one in detail in the sequel.

2.1 Trace generation

The first step of our approach is to generate simulated power traces in a noise-free environment.

Implementation. To accomplish this, the masking scheme is typically implemented in a high-level software language. The implementation is meant to generically reproduce the intermediate values present in the masking scheme, and can be typically written from the pseudo-code description of the masking scheme. (Alternatively, the description of the masking scheme can be tailored to a specific software or hardware implementation and incorporate details from those.)

Execution. This implementation is executed many times, and during each execution, the instrumentation environment observes each variable V that the implementation handles at time n. At the end of each execution, the environment emits a leakage trace $c[n]$. Each time sample n within this trace consists of leakage $L(V)$ of the variable V handled at time n. The leakage function L is predefined; typical instantiations are the Hamming weight, the least significant bit, the so-called zero-value model or the identity function.

Randomness. The high-level implementation may consume random numbers (for example, for remasking.) This randomness is provided by a PRNG.

2.2 Trace pre-processing

This step is only executed if the masking scheme claims higher-order security. The approach is similar to higher-order DPA attacks [CJRR99] and higher-order leakage detection [SM15]. Suppose the scheme claims security at order k. We pre-processes each simulated trace $c[n]$ to yield $c'[n_1, \ldots, n_k]$ through a combination function as

$$c'[n_1, \ldots, n_k] = \prod_{i=1}^{i=k} (c[n_i] - \bar{c}[n_i]). \tag{1}$$

The result is a preprocessed trace c'. The length of the trace is expanded from N to $\binom{N}{k}$ unique time samples. (It is normally convenient to treat c' as a uni-dimensional trace.)

2.3 Leakage detection

The next step of our approach is to perform a leakage detection test on the (potentially pre-processed) simulated traces. In its simplest form, a leakage detection test [CKN00, CNK04, GJJR11, CDG+13, SM15] tries to locate and potentially quantify information leakage within power traces, by detecting statistical dependencies between sensitive data and power consumption. In our context, if the test detects information leakage on the simulated traces, this means that the masking scheme fails to provide the promised security guarantees.

Procedure. The instrumentation environment performs a fixed-vs-fixed leakage detection test using the T-test distinguisher [CDG+13].

The process begins by simulating a set of power traces with fixed unmasked intermediate $z = z_0$ and another set of traces with different unmasked intermediate value $z = z_1$. Typical choices for the intermediate z are the full unmasked state or parts of it. Then, a statistical hypothesis test (in this case, T-test) is performed per time sample for the equality of means. The T-test [Stu08, Wel47] first computes the following statistic

$$t[n] = \frac{m_0[n] - m_1[n]}{\sqrt{\frac{s_0^2[n]}{N_0} + \frac{s_1^2[n]}{N_1}}} \tag{2}$$

where $m_i[n]$, $s_i^2[n]$, N_i are respectively the sample mean, variance and number of traces of population $i \in \{0, 1\}$ and n is the time index. This statistic $t[n]$ is compared against a predefined threshold C. A common choice is $C = \pm 4.5$, corresponding to a very high statistical significance level of $\alpha = 0.001$. If the statistic $t[n]$ surpasses the threshold C, the test determines that the means of the two distributions are significantly different, and thus the mean power consumption

of (potentially pre-processed) simulated power traces carry information on the intermediate z. In this case, we say that the masking scheme exhibits leakage at time sample n and flunks the test. Otherwise, if no leakage is detected, another test run is executed with different specific values for z_0 and z_1. The test is passed only if no leakage is detected for any value of z_0 and z_1. (Typically, there are only a couple dozen of (z_0, z_1) pairs if the optimizations described in the next section are applied.) Note that a time sample n may correspond to a single variable (first-order leakage) or a combination of variables (higher-order leakage), if a pre-processing step is executed.

On fixed-vs-fixed. Using fixed-vs-fixed instead of fixed-vs-random has the advantage of faster convergence of the statistic (at the expense of leakage behavior assumptions that are benign in our context). (This has been previously observed by Durvaux and Standaert [DS16] in a slightly different context.) One could also use a fix-vs-random test. This usually results in a more generic evaluation.

2.4 Optimizations

We note that the following "engineering" optimizations allow to lower the computational complexity so that it is becomes very fast to test relevant masking schemes.

Online algorithms. There is certainly no need to keep in memory the complete set of simulated power traces. For the computation of the T-test as Eq. 2, one can use online formulas to compute means and variances present in the formula. These algorithms traverse only once through each trace, so that a simulated power consumption trace can be generated, processed and thrown away. This makes the memory consumption of the approach independent of the number of traces used. More number of traces would require just more computational time, but not more memory. We note that the same is possible in higher-order scenarios. Lengthy but straightforward calculations show that a T-test on pre-processed traces can be computed online using well-known online formulae for (mixed) higher-order moments [PÓ8]. (This was previously reported by Schneider and Moradi [SM15].)

Scale down the masking scheme. It is usually possible to extrapolate the masking scheme to analogous, trimmed down, cryptographic operations that work with smaller bit-widths or smaller finite fields. For example, when masking the AES sbox, many masking schemes [RP10, CPRR13] rely on masked arithmetic (masked multiplication and addition blocks) in $\mathrm{GF}(2^8)$ to carry out the inversion in $\mathrm{GF}(2^8)$. It is often convenient to scale down the circuit to work on, say, $\mathrm{GF}(2^4)$ for testing purposes –since the masking approach normally does not rely on the specific choice

of field size, any flaw exhibited in the smaller GF(2^4) version is likely to be exhibited in the GF(2^8) version of the algorithm (and vice versa). By experience we have observed that statistical biases tend to be more pronounced in smaller field sizes, and thus are more easily detectable. (See for instance [PRR14].) We suggest the use of this heuristic whenever possible for an early alert of potential problems.

Reduce the number of rounds. There is little sense to check for a first-order leak in more than a single round of an iterative cryptographic primitive, such as AES. If the implementation is iterative, any first-order flaw is likely to show up in all rounds. When testing for higher order security, however, one should take into account that the flaw may appear from the combination of variables belonging to different rounds.

Judiciously select the components to check. For first-order security it is sufficient to check each component of the masking scheme one by one in isolation. The situation is slightly different in the multivariate scenario, where multiple components can interfere in a way that degrades security. Still, the practitioner can apply some heuristics to accelerate the search, such as testing for second-order leakage first only in contiguous components. For example, second-order leakage is likely to appear earlier between two variables within the same round or belonging to two consecutive rounds.

Deactivate portions of the plaintext. To accelerate the leakage search, a substantial portion of the plaintext can be deactivated, that is, fixed to a constant value or even directly taken out from the algorithm. For example, in the case of AES-128 one could deactivate 3 columns of the state, test only 4 active plaintext bytes and still test for the security of all the components within one round.

Carefully fix the secret intermediate values. As we described, the framework fixes two values z_0, z_1 for the unmasked sensitive intermediate, and then compares the simulated traces distributions conditioned on z_0 and z_1. Depending on the algorithm, concrete choices for z_i (such as fixed points of the function being masked) can produce "special" leakage. For example, in AES if we choose z_1 such that the input to the inversion is `0x00`, we can hit faster zero-value type flaws.

3 Results

In this section we provide experimental results. We first begin by testing the first-order security claim of two schemes, one that fails the claim (Section 3.1) and

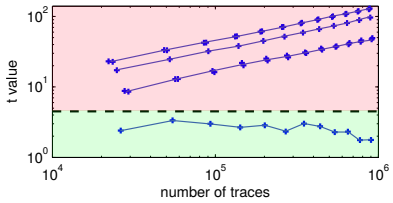


Figure 1 – T-statistic (absolute values) of the IP masking scheme, under a HW leakage model. Deemed insecure (clearly exceeds the threshold at $t = 4.5$.)

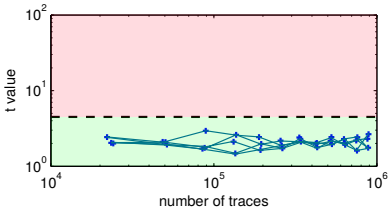


Figure 2 – T-statistic (absolute values) applied to the Coron table recomputation masking scheme, under an Identity leakage model. First order test. Deemed secure (no value beyond the threshold at $t = 4.5$.)

another that fulfills it (Section 3.2). Then we will focus on second- and third- order claims (Section 3.3 and 3.4 respectively). We point out a new second-order flaw in Section 3.5, we elaborate on how previously published flaws were discovered in Section 3.6. Finally in Section 3.7 we report on the use of the tool when designing masked circuits.

3.1 Smoke test: reproducing a first-order flaw

As a first test, we test the first-order security of the scheme published in [BFGV12]. We will refer to this scheme as IP in the sequel. We focus on reproducing the results from [PRR14],

Test fixture. We study first the `IPRefresh` procedure. This procedure performs a refreshing operation on the input IP shares. We scale down the scheme to work in GF(2^2) following Section 2.4. The instrumentation framework finds 141 intermediate variables within a single execution of `IPRefresh`. The chosen leakage function is Hamming weight, and there is no pre-processing involved.

Leakage detection. We ran the $\binom{4}{2} = 6$ possible fixed-vs-fixed tests covering all possible combinations of pairs of different unshared input values (z_1, z_0). (Here z_i is the input to `IPRefresh`.) For each test, the maximum absolute t-value, across all time samples, is plotted in the y-axis of Fig. 1 as a function of the number of

simulated traces (x-axis). A threshold for the T-test at 4.5 is also plotted as a dotted line. This threshold divides the graph into two regions: a t-statistic greater than $|C| = 4.5$ (in red) means that the implementation fails the test, while a t-statistic below 4.5 (area in green) does not provide enough evidence to reject the hypothesis that the scheme is secure. We can see that 5 out of 6 tests clearly fail in Fig. 1, since they attain t-values around 100 greater than C. Thus, the `IPRefresh` block is deemed insecure. (Similar observations apply to the `IPAdd` procedure.)

It is also possible to appreciate the nature of the T-test statistic: the t-statistic grows with the number of traces N as of $\sqrt{N}$ in the cases that the implementation fails the test (note that the y-axis is in logarithmic scale.) This can be interpreted as follows: as we have more measurements, we build more confidence to reject the null hypothesis (in our context being that the masking is effective.) If the number of simulated traces is large enough and no significant t-value has been observed, the practitioner can gain confidence on the scheme not being flawed. We will find this situation in the next subsection and elaborate on this point.

3.2 A first-order secure implementation

We tested the table recomputation scheme of Coron [Cor14]. This scheme passes all fixed-vs-fixed tests with the identity leakage model. The results are plotted in Figure 2. We can observe that the t-statistic never crosses the threshold of 4.5 for any test, and thus we cannot reject the null hypothesis that the implementation is secure (i.e., the implementation is deemed secure, "*on the strength of the evidence presented*" [CKN00].) Although it is theoretically possible that the masking scheme exhibits a small bias that would only be detectable when using more than 10^6 traces, that flaw would be negligible from a practical point of view when using $\leq 10^6$ traces, and definitely impossible to exploit in a noisy environment if it is not even detectable at a given trace count, in a noiseless scenario.

3.3 Reproducing a second-order flaw

To show that our proposed tool can also detect higher-order flaws, we implemented the scheme of Rivain and Prouff (RP) from [RP10]. For the allegedly second-order secure version of this scheme, there is a second-order flaw as spotted by Coron et. al. in [CPRR13] between two building blocks: `MaskRefresh` and `SecMult`. We will see that we can easily spot this flaw with the methodology proposed in this paper.

Text fixture. We implemented the second-order masked inversion $x \mapsto x^{-1}$ in $\mathrm{GF}(2^n)$ as per [RP10] with $n = 3$. This inversion uses the `MaskRefresh` and `SecMult` procedures. In this case, we enable second-order pre-processing (on the

```
 70 void MaskRefresh(u8 *s) {
 71  u8 r;
 72  for (int i = 1; i < number_shares; i++) {
 73   r = rnd ();
 74   s[0] ^= r;
 75   s[i] ^= r;
 76  }
 77 }
...
110 void SecMult (u8 *out, u8 *a, u8 *b) {
111  u8 aibj,ajbi;
...
114  for (int i = 0; i < number_shares; i++) {
115   for (int j = i + 1; j < number_shares; j++) {
...
119    aibj = mult(a[i], b[j]);
120    ajbi = mult(a[j], b[i]);
-------------------------------------------------
$ ./run
entering fixed_vs_fixed(00,01)
> leakage detected with 1.20k traces
  higher order leakage between
    line 74 and
    line 120
  with tvalue of -7.03
```

Figure 3 – Excerpts of the code and output of the leakage detection for the RP scheme.

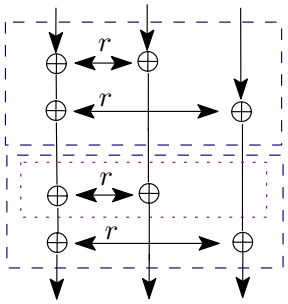


Figure 4 – Two `MaskRefresh` concatenated. As explained in the text, the second refresh can be optimized to reduce the randomness requirements yet still achieving second order security.

fly), expanding 135 time samples to $\binom{135}{2} = 9045$ time samples. Some excerpts of the implementation are shown in Fig. 3, top.

Results. The instrumentation frameworks takes less than 5 seconds to determine that there is a second order leakage between the variable handled at line `74` (inside `MaskRefresh`) and `120` (inside `SecMult`), as Fig. 3, bottom, shows. Note that it is trivial to backtrack to which variables corresponds a leaking time sample, and thus determine the exact lines that leak jointly.

Fixing the second-order flaw. The folklore solution to fix the previous second-order flaw is to substitute each `MaskRefresh` module by two consecutive `MaskRefresh` invocations, as shown in Fig. 4. Applying the leakage detection tests to this new construction shows that the leak is effectively gone. However, it is quite reasonable to suspect that this solution is not optimal in terms of randomness requirements. We can progressively strip down this design by eliminating some of the randomness of the second refreshing and check if the design is still secure. We verified in this very simple test fixture that if we omit the last randomness call (that is, we only keep the dotted red box instead of the second dashed box in Fig. 4), the higher-order leaks are no longer present.

3.4 Reproducing a third order flaw

Schramm and Paar published at CT-RSA 2006 [SP06] a masked table lookup method for Boolean masking claiming higher-order security. This countermeasure was found to be flawed by Coron et al. at CHES 2007. Coron et al. found a third-order flaw irrespective of the security parameter of the original scheme. We reproduced their results by setting $k = 3$ when preprocessing the traces as in Eq. 1.

The flaw of [CPR07] was detected in less than one second, which demonstrates that the tool is also useful to test the higher-order security claims of masking schemes.

3.5 Schramm–Paar second-order leak

Here we report on a new second-order flaw that we found with the presented tool in the masked table recomputation method of Schramm and Paar when used with unbalanced sboxes.

Schramm–Paar method. The goal of the masked table recomputation is to determine the sbox output shares $N_0, N_1, \ldots, N_d$ from the sbox input shares $M_0, M_1, \ldots, M_d$. Schramm–Paar proceed as follows (we borrow the notation from [CPR07]):

1. Draw d output shares $N_1, \ldots, N_d$ at random.
2. Compute from $N_1, \ldots, N_d$ a table S^* such that

$$S^*(x) = S\left(x \oplus \bigoplus_{i=1}^{d} M_i\right) \oplus \bigoplus_{i=1}^{d} N_i \tag{3}$$

3. Set $N_0 := S^*(M_0)$

We set here $d = 2$, and aim at second-order security. An important part of the procedure is to build the table S^* in a way that the higher-order security claims are fulfilled. [SP06] proposes several methods. However, for the purposes of this paper the details of the recomputation method are not important.

Test fixture. Following the guidelines of Section 2.4, we implement a very stripped down version of the table recomputation method. We fix the simplest unbalanced sbox $S = (0, 0, 0, 1)$ (an AND gate), and work with 2-bit inputs and outputs leaking Hamming weights. In a couple of seconds the tool outputs 4 different bivariate second-order leakages, corresponding to the pairs $(S^*(i), N_0)$ for each i in the domain of S^*. Here $S^*(i)$ is the i-th entry on the S^* table, and N_0 is one output mask.

Once these leaks are located, proving them becomes an easy task. And also it becomes easy to generalize and see that the flaw appears whenever S is unbalanced. (We verified that second-order attacks using the leakage of $S^*(0)$ and N work as expected.)

3.6 Higher-order threshold implementations

Here we report on how the observations from [RBN+15] regarding the security of higher-order threshold implementations [BGN+14] were found. The results of this section are obviously not new; the focus here is on the methodology carried out to find those.

Intuition. The first suspicion stems from the fact that higher-order threshold implementations originally claimed that the composition of sharings provides higher-order security, if the sharings satisfy some property, namely uniformity. This is a very surprising result, since it would imply that there is no need to inject fresh randomness during the computation, minimizing overheads. In contrast, all other previously published higher-order masking schemes need to inject randomness from time to time as the computation progresses. For example, the security proof of private circuits (one of the earliest masking schemes) [ISW03] critically relies on the fresh randomness to provide security.

Test fixture. The hypothesis is that the previous security claim does not hold, that is, the concatenation of uniform sharings do not provide higher-order security. To test this, we design a minimal working test fixture consisting of a 32-round Feistel cipher with a blocksize of 4 bits. For more details see [RBN+15]. The shared version aims at providing second-order security, and shares each native bit into 5 shares. The traces consist of 225 "timesamples" (each one comprising one leaked bit, including initialization.) This spans to 25650 timesamples after second-order pre-processing.

Cranking it up. We run the simulation for a night (about 8 hours), having simulated 200 million traces. We performed a fixed-vs-fixed test with unshared initial state `0000` and `1111`. (There is no key in this cipher, the initial state is considered to be the secret.) (This is grossly unoptimized code.) The results of the leakage detection test is drawn in Figure 5. We plot on the x- and y-axes the round index, and each pixel in red if the t statistic surpasses the value 80, green otherwise. We can see that many pairs of rounds leak jointly, in contradiction with the security claims of the scheme. In Figure 6 the same information is plotted but changing the threshold to $|t| > 5$. We can see, surprisingly, that *almost all* pairs of rounds lead to second-order leakage. A bit of manual mechanical effort is required to prove this, but not more than taking a covariance.

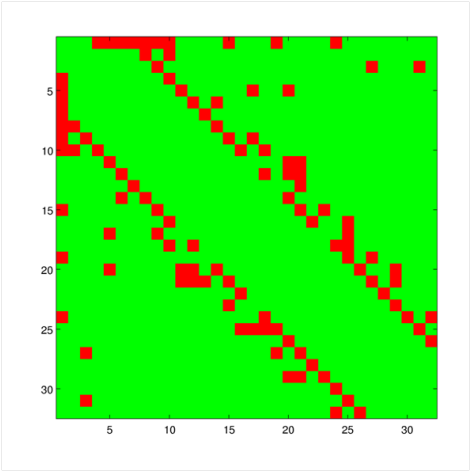


Figure 5 – Pairs of rounds with $|t| > 80$

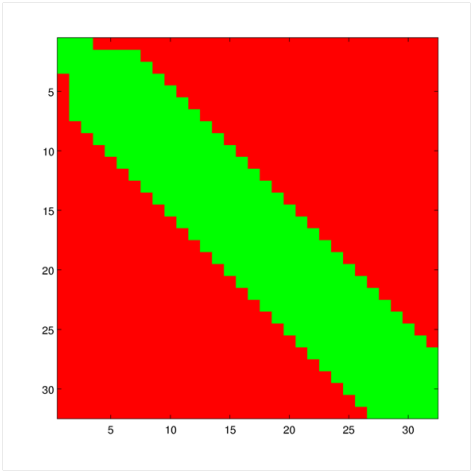


Figure 6 – Pairs of rounds with $|t| > 5$

3.7 Refreshing in higher-order threshold AES sbox

The designers from [CBR+15] had access to the tool presented in this paper. They performed several design iterations, and verified the design on each iteration. The final evaluation was performed on an FPGA.

Text fixture. We implemented the whole sbox, with no downscaling of the components to work in smaller fields. We leak register bits and the input value (identity leakage function) to combinatorial logic blocks. (This is to account for glitches as will be explained below.)

First-order leaks. Within one day, a first-order leak was identified due to a design mistake. This design error considered the concatenation of two values $a||b$ as input to the next stage; each value a and b considered independently is a uniform sharing but its concatenation $a||b$ is not, and hence the first order leak. This first-order leak disappears if a refresh is applied to the inputs of one $\mathrm{GF}(2^2)$ multiplier using 4 units of randomness (here 1 unit = 1 random field element = 2 bits). This refresh block is similar to the 2010 Rivain–Prouff refresh block [RP10], we remind it uses $n-1$ units of randomness to refresh n shares (in our particular case here $n = 5$). We will see later that this refresh is problematic in the higher-order setting.

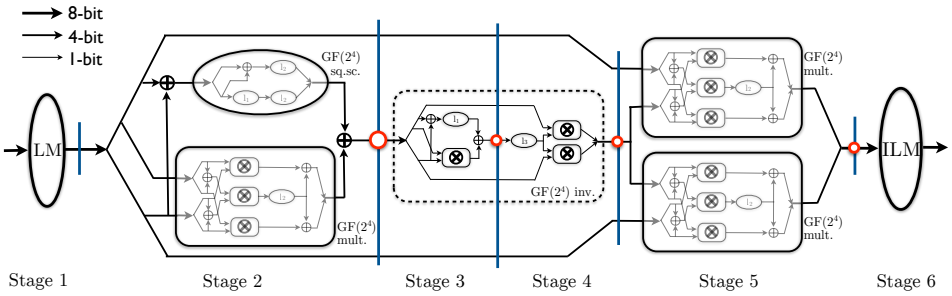


Figure 7 – Higher-order masked AES sbox from de Cnudde et al.

Second-order leaks. Subsequently, two second-order bivariate leaks were identified between register values. This was solved by placing a refresh block between stage 2 and 3 from Figure 7 (taken from [CBR+15]).

In addition, many second-order bivariate leaks were identified between input values to combinatorial logic blocks. In theory, hardware glitches could express these leakages. Those disappear if one uses a "full refresh" using 5 units of randomness. This effect was previously observed [BBD+15, RBN+15] and is a reminiscent of [CPRR13].

Other uses. We also used a preliminary version of this tool in [RRVV15].

4 Discussion

4.1 Implementing the framework

We implemented the instrumentation framework on top of `clang`-LLVM. The whole implementation (including leakage detection code) takes around 700 lines of C++ code, which shows the inherent simplicity of our approach. It is easy to audit and maintain.

4.2 Time to discover flaw, computational complexity and scaling.

The computational requirements of the proposed approach are very low. In Fig. 8 we write the elapsed execution times required to spot the flaws from Sections 3.1,

3.3 and 3.4. We can see that the flaws were identified in a matter of seconds on a standard computer. All the experiments on this paper were carried out on a modest 1.6 GHz laptop with 2 GB of RAM.

Bottlenecks. The main bottleneck on the running time of the methodology is the first step: trace generation. The RP scheme is the one that took longer to detect the flaw (5 seconds), presumably because of two reasons: a) the scheme is more inherently complex and thus it takes more time to simulate each trace and b) the bias exhibited in the scheme is smaller than the bias of other schemes, and thus more traces are required to detect such a bias. We note that no special effort on optimizing the implementations was made, yet, an average throughput of 5k trace per second (including instrumentation) was achieved. The overhead of instrumentation in the running time was estimated to make the implementation on average $\approx \times 1.6$ slower.

Time to pass. The time it takes to discover a flaw is normally less than the time it takes to deem a masking scheme secure. For example, to assess that the patch of Section 3.3 is indeed correct, it took about 6 minutes to perform a fix-vs-fix test with up to 1 million traces (no leakage was detected). All possible 6 tests take around 37 minutes. (The threshold of 1 million traces was chosen arbitrarily in this example.)

Parallelization. We remark that this methodology is embarrassing parallel. Thus, it is much easier to parallelize to several cores or machines than other approaches based on SAT.

Memory. The memory requirements for this method are also negligible, taking less than 4.5 MB of RAM on average. More interestingly, memory requirements are constant and do not increase with the number of simulated traces, thanks to online algorithms.

Scaling. The execution time of our approach scales linearly with the number of intermediates when testing for first-order leakage, quadratically when testing for second-order leakage and so on. This scaling property is exactly the same as for DPA attacks. We could benefit from performance improvements that are typically used to mitigate scaling issues in DPA attacks such as trace compression, but did not implemented those yet.

Scheme	Flaw order	Field size	Time	Traces needed
IP	1	4	0.04s	1k
RP	2	4	5s	14k
SP	3	4	0.2s	2k

Figure 8 – Running time to discover flaw in the studied schemes, and number of traces needed to detect the bias.

4.3 Limitations

Risk of false negatives. Our tool should not be taken as the only test when assessing a masked implementation, and is not meant to substitute practical evaluation with actual measurements. Our tool provides an early warning that the masking scheme may be structurally flawed, "by design". However, even when the masking scheme is theoretically secure, it is still possible to implement it in an insecure way. This will not be detected with the proposed tool. For example, in the case of a first-order masked software implementation, an unfortunate choice of register allocation may cause distance leakage between shares, leading to first-order leakage. Without register allocation information, our tool will not detect this issue. One could provide this kind of extra information to our tool. We left this as future work.

4.4 Related works

There are already some publications that address the problem of automatic verification of power analysis countermeasure.

SAT-based. Sleuth [BRNI13] is a SAT-based methodology that outputs a hard yes/no answer to the question of whether the countermeasures are effective or not. A limitation of [BRNI13] is that it does not attempt to quantify the degree of (in)security. A first approximation to the problem was tackled in [EWTS14, ABMP13].

MiniCrypt-based. Barthe et al. [BBD+15] use program verification techniques to build a method prints a proof of security for a given masking scheme. It is very hard to compare our tool with theirs since they are fundamentally different. The goal is also different: while our results are probabilistic, the goal of Barth et al. is to categorically prove the security of the scheme. Depending on the context, one might be preferrable over the other. The two approaches are also very different. Barthe

et al. base their approach on EasyCrypt, a sophisticated "toolset for reasoning about relational properties of probabilistic computations with adversarial code."

Considerations related to other approaches. While our approach does certainly not carry the beauty of proofs and formal methods, it offers a very practice-oriented methodology to test the soundness of masking schemes. Our approach is in nature statistical, and is a necessary condition for a masked scheme to be sound. It can be thought of a worst-case scenario, where the adversary has access to noiseless and sychronized traces. A more formal study can then be performed with the methods of Barthe et al. to gain higher confidence, since the output of the tool of Barthe et al. is a hard proof.

4.5 Which leakage function to select?

In previous Section 2 we mentioned that the practitioner has to choose a leakage function to generate the simulated traces. It turns out that the specific choice of leakage function seems not to be crucial —any reasonable choice will work. Figure 9 compares different leakage functions: Hamming weight, identity, least-significant bit and zero-value. The test fixture is the same one as in Section 3.1. For each leakage function, we performed all possible fixed-vs-fixed tests. Figure 9 is composed of 4 plots, one per leakage function. We can see that for any leakage function, there is at least one fixed-vs-fixed test that fails. For the identity leakage function, *all* tests fail. Thus, it is often convenient to use it to detect flaws faster (more fixed-vs-fixed tests fail.) We speculate that this behavior may depend on the concrete masking method used, and leave a detailed study as future work.

Glitches and identity leakage. We note that we can include the effect of hardware glitches in our tool. Note that the information leaked by a combinatorial logic block F on input x due to glitches is contained already in the input x. Thus, we can simulate the information leaked by hardware glitches, even if we do not have a precise timing model of the logic function, by leaking the whole input value x (that is, using the identity leakage model.)

This would correspond to an extremely glitchy implementation of F where glitches would allow to observe the complete input. This is certainly a worst-case scenario. Crucially, glitches would not reveal *more* information than x. This trick of using the identity leakage model on inputs of combinatorial blocks is helpful when evaluating, for example, masked threshold implementations.

Another alternate approach is to add a detailed gate-level timing model to simulate glitches. If such timing model is available, the detection quality can be substantially enhanced.

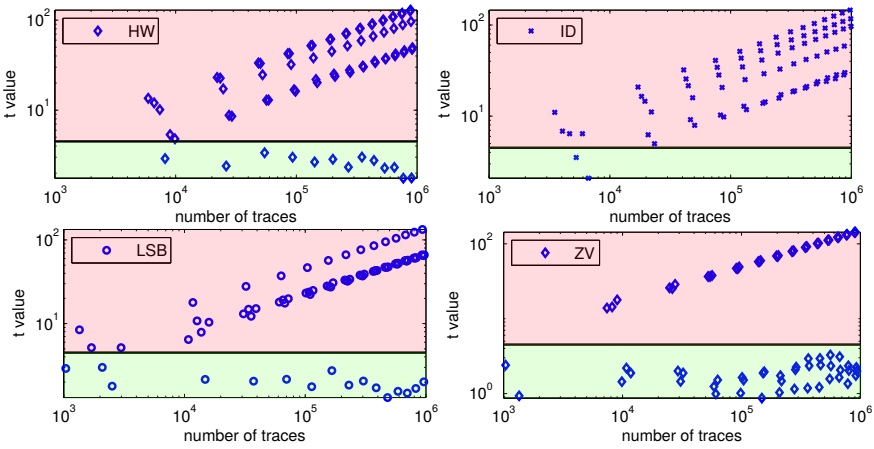


Figure 9 – Influence of leakage function.

5 Conclusion

We described a methodology to test in an automated way the soundness of a masking scheme. Our methodology enjoys several attractive properties: simplicity, speed and scalability. Our methodology is based on established and well-understood tools (leakage detection). We demonstrated the usefulness of the tool by detecting state-of-the-art flaws of modern masking designs in a matter of seconds with modest computational resources. In addition, we showed how the tool can assist the design process of masked implementations.

Acknowledgements. We thank an anonymous reviewer that found a mistake in Section 3.5, François-Xavier Standaert for extensive comments and Ingrid Verbauwhede. The author is funded by a PhD fellowship of the Fund for Scientific Research - Flanders (FWO). This work was funded also by Flemish Government, FWO G.0550.12N, G.00130.13N, Hercules Foundation AKUL/11/19, and through the Horizon 2020 research and innovation programme under grant agreement 644052 HECTOR.

References

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded*

Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, 2002.

[ABMP13] Giovanni Agosta, Alessandro Barenghi, Massimo Maggi, and Gerardo Pelosi. Compiler-based side channel vulnerability analysis and optimized countermeasures application. In *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pages 81:1–81:6. ACM, 2013.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 457–485. Springer, 2015.

[BFGV12] Josep Balasch, Sebastian Faust, Benedikt Gierlichs, and Ingrid Verbauwhede. Theory and Practice of a Leakage Resilient Masking Scheme. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 758–775. Springer, 2012.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2014.

[BRNI13] Ali Galip Bayrak, Francesco Regazzoni, David Novo, and Paolo Ienne. Sleuth: Automated verification of software power analysis countermeasures. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 293–310. Springer, 2013.

[CBR+15] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the AES s-box. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS*

2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.

[CDG+13] Jeremy Cooper, Elke DeMulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) methodology in practice. International Cryptographic Module Conference, 2013.

[CGPR08] Jean-Sébastien Coron, Christophe Giraud, Emmanuel Prouff, and Matthieu Rivain. Attack and improvement of a secure s-box calculation based on the fourier transform. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 1–14, 2008.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[CKN00] Jean-Sébastien Coron, Paul C Kocher, and David Naccache. Statistics and Secret Leakage. In Yair Frankel, editor, *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, volume 1962 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2000.

[CNK04] Jean-Sébastien Coron, David Naccache, and Paul C Kocher. Statistics and secret leakage. *ACM Trans. Embedded Comput. Syst.*, 3(3):492–508, 2004.

[Cor14] Jean-Sébastien Coron. Higher Order Masking of Look-Up Tables. In Phong Q Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 441–458. Springer, 2014.

[CPR07] Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side Channel Cryptanalysis of a Higher Order Masking Scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 28–44. Springer, 2007.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-Order Side Channel Security and Mask Refreshing. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[DS16] François Durvaux and François-Xavier Standaert. From improved leakage detection to the detection of points of interests in leakage traces. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 240–262. Springer, 2016.

[EWTS14] Hassan Eldib, Chao Wang, Mostafa M. I. Taha, and Patrick Schaumont. QMS: evaluating the side-channel resistance of masked software from source code. In *The 51st Annual Design Automation Conference 2014, DAC '14, San Francisco, CA, USA, June 1-5, 2014*, pages 1–6. ACM, 2014.

[GJJR11] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. A testing methodology for side channel resistance validation. NIST non-invasive attack testing workshop, 2011.

[GP99] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private Circuits: Securing Hardware against Probing Attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[PÓ8] Philippe Pébay. Formulas for robust, one-pass parallel computation of co- variances and arbitrary-order statistical moments. Technical Report SAND2008-6212, Sandia National Laboratory, 2008.

[PGA06] Emmanuel Prouff, Christophe Giraud, and Sébastien Aumônier. Provably secure s-box implementation based on fourier transform. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 216–230. Springer, 2006.

[PRR14] Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. On the Practical Security of a Leakage Resilient Masking Scheme. In Josh Benaloh, editor, *Topics in Cryptology - CT-RSA 2014 - The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, volume 8366 of *Lecture Notes in Computer Science*, pages 169–182. Springer, 2014.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating Masking Schemes. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[RP10] Matthieu Rivain and Emmanuel Prouff. Provably Secure Higher-Order Masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-lwe implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 495–513. Springer, 2015.

[SP06] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[Stu08] Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.

[Wel47] Bernard L Welch. The generalization of Student's problem when several different population variances are involved. *Biometrika*, pages 28–35, 1947.

Curriculum Vitae

Oscar Reparaz was born on December 25th in Madrid, Spain. He received the Master's degree in Electrical Engineering from Universidad Politécnica de Madrid (ETSIT/UPM) in June 2011.

In September 2011, he joined the COSIC research group at the Department of Electrical Engineering (ESAT) of the KU Leuven. His research has been generously funded by a personal grant from the Fund for Scientific Research, Flanders (FWO). He was an intern at Square, Inc. during summer 2014 in San Francisco, US.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
COMPUTER SECURITY AND INDUSTRIAL CRYPTOGRAPHY
Kasteelpark Arenberg 10 box 2452
B-3001 Heverlee